# NAVAL POSTGRADUATE SCHOOL
## Monterey, California


# THESIS

**DEVELOPING A STANDARD PLATFORM-LEVEL ARMY OBJECT MODEL**

by

Douglas E. Dudgeon

September 1997

Thesis Advisor: Arnold H. Buss

**Approved for public release; distribution is unlimited.**

# REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188



| 1. AGENCY USE ONLY (Leave Blank) | 2. REPORT DATE | 3. REPORT TYPE AND DATES COVERED |
|---|---|---|
| | September 1997 | Master's Thesis |

**4. TITLE AND SUBTITLE**
DEVELOPING A STANDARD PLATFORM-LEVEL ARMY OBJECT MODEL

**5. FUNDING NUMBERS**

**6. AUTHOR(S)**
Dudgeon, Douglas E.

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**
Naval Postgraduate School
Monterey, CA 93943-5000

**8. PERFORMING ORGANIZATION REPORT NUMBER**

**9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)**
U.S. Army TRADOC Analysis Center
P. O. Box 8692
Monterey, CA 93943-0692

**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

**11. SUPPLEMENTARY NOTES**
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

**12a. DISTRIBUTION / AVAILABILITY STATEMENT**
Approved for public release; distribution is unlimited

**12b. DISTRIBUTION CODE**

**13. ABSTRACT** (Maximum 200 words)

Before 1990, the Department of Defense (DoD) modeling and simulation effort was fragmented and uncoordinated. Developers of new simulations usually had to start from scratch and could reuse very few of the components from legacy models. Simulations from different developers were incompatible and inconsistent. One of the features of object-oriented programming (OOP) is the prospect of reusing design and code on future projects. However, reuse does not simply happen, it must be planned by thinking beyond the immediate application and developing a more general design.

Interoperability and reuse are limited because DoD lacks a common technical framework for simulation architecture. The Army Modeling and Simulation Office (AMSO) Master Plan's primary objective is the creation of this framework. Central to the plan is the development of a standard army object model. This thesis documents the development of the initial version of the standard army object model. The role of the standard army object model is to enhance interoperability and reuse and to achieve a minimal level of uniformity in Army simulations. This standard will specify object-oriented properties for classes, and class hierarchies for use with future high resolution simulation development.

A modified version of Rumbaugh's Object Modeling Technique was used to develop the object model. A component-based design was adopted. The object model is code independent and minimal in design to allow developers maximum flexibility. The research indicates that the standard army object model can also serve as a focal point for other initiatives outlined in the AMSO Master Plan.

| 14. SUBJECT TERMS | 15. NUMBER OF PAGES |
|---|---|
| Object modeling, Rumbaugh's OMT, Object Oriented Analysis, simulation, combat modeling | 85 |
| | 16. PRICE CODE |

| 17. SECURITY CLASSIFICATION OF REPORT | 18. SECURITY CLASSIFICATION OF THIS PAGE | 19. SECURITY CLASSIFICATION OF ABSTRACT | 20. LIMITATION OF ABSTRACT |
|---|---|---|---|
| Unclassified | Unclassified | Unclassified | UL |

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18
298-102

**Approved for public release; distribution is unlimited.**

# DEVELOPING A STANDARD PLATFORM-LEVEL ARMY OBJECT MODEL

Douglas E. Dudgeon
Captain, United States Marine Corps
B.S., United States Naval Academy, 1991

Submitted in partial fulfillment
of the requirements for the degree of

**MASTER OF SCIENCE IN OPERATIONS RESEARCH**

from the
**NAVAL POSTGRADUATE SCHOOL**
**September 1997**

Author: Douglas E. Dudgeon
Douglas E. Dudgeon

Approved by:
Arnold Buss, Thesis Advisor

Leroy A. Jackson
Leroy A. Jackson, Second Reader

Richard E. Rosenthal
Richard E. Rosenthal, Chairman
Department of Operations Research

## ABSTRACT

Before 1990, the Department of Defense (DoD) modeling and simulation effort was fragmented and uncoordinated. Developers of new simulations usually had to start from scratch and could reuse very few of the components from legacy models. Simulations from different developers were incompatible and inconsistent. One of the features of object-oriented programming (OOP) is the prospect of reusing design and code on future projects. However, reuse does not simply happen, it must be planned by thinking beyond the immediate application and developing a more general design.

Interoperability and reuse are limited because DoD lacks a common technical framework for simulation architecture. The Army Modeling and Simulation Office (AMSO) Master Plan's primary objective is the creation of this framework. Central to the plan is the development of a standard army object model. This thesis documents the development of the initial version of the standard army object model. The role of the standard army object model is to enhance interoperability and reuse and to achieve a minimal level of uniformity in Army simulations. This standard will specify object-oriented properties for classes, and class hierarchies for use with future high resolution simulation development.

A modified version of Rumbaugh's Object Modeling Technique was used to develop the object model. A component-based design was adopted. The object model is code independent and minimal in design to allow developers maximum flexibility. The research indicates that the standard army object model can also serve as a focal point for other initiatives outlined in the AMSO Master Plan.

# TABLE OF CONTENTS

# TABLE OF FIGURES

## EXECUTIVE SUMMARY

The Department of Defense has used simulations for force and weapons design, material acquisition and training for more than four decades. Before 1990, the Department of Defense (DoD) modeling and simulation (M&S) efforts were fragmented and uncoordinated. Developers of new simulations typically had to start from scratch and were able to reuse few of the components from legacy models. This was mostly due to poor documentation and the fact that many of the models were implemented in procedural languages. Developers used their own data and algorithms in their simulations. Consequently, simulations from different developers were usually incompatible and yielded inconsistent portrayals of the same phenomenon.

Object-oriented programming (OOP) has replaced procedural programming as the dominant programming paradigm. Object-oriented programming is a style of software development which models the relationships among the objects which make up a problem rather than the procedures used to solve the problem. One of the features of OOP is the prospect of reusing design and code for future projects. This saves time and money and allows more effort to be placed on verifying the code that is reused. However, reuse does not simply happen, it must be planned by thinking beyond the immediate application and developing a more general design.

Interoperability is the degree to which different simulations interact. Interoperability and reuse have been limited in the past, because DoD lacked a common technical framework for simulation architecture. The Defense Modeling and Simulation Office (DMSO) was created to coordinate simulation policy for the DoD. Based on

guidance from DMSO, the Army Modeling and Simulation Office (AMSO) issued the AMSO Master Plan to establish the Army's vision for modeling and simulation. There are six major objectives in the AMSO Master Plan. The primary objective is to develop a common technical framework. The standard army object model is part of this effort to implement a common technical framework. [AMSO, 1995]

An object model is a representation of the static, structural aspects of a system. Developing an object model is often the first step in writing an object oriented program. The role of the standard army object model is to enhance interoperability and reuse and to achieve a minimal level of uniformity in Army simulations. This standard will specify object-oriented properties for classes, and class hierarchies for use with future high resolution simulation development. This will provide common names and interfaces through which objects can communicate. These classes should serve as a common starting point for future development efforts. The only constraint that the object model imposes is that the programming language used to implement the simulation must support object-oriented programming.

This thesis documents the development of the initial version of the standard army object model. The developed object model is code independent and minimal in design to allow developers maximum flexibility. A component-based modeling philosophy is adopted. Each component is used to represent one or more battlefield functions such as target acquisition, attrition and sustainment.

The research indicates that a the standard army object model can also serve as a focal point for other initiatives outlined in the AMSO Master Plan. Two of these initiatives are the development of standard algorithms and a Functional Description of the

Battlespace (FDB). Standard algorithms are standard descriptions of primary battle field functions such as attrition and target detection. The FDB is a simulation independent repository of data designed to support simulation development. The standard algorithms can be incorporated into the object model allowing design and data requirements for the algorithms to be easily displayed. Thus the standard army object model serves as an effective interface between standard algorithms and the data required to support those algorithms.

# I. INTRODUCTION

The United States Army has used computer simulations for years to train combat leaders and to analyze problems in areas such as combat operations, procurement, force structure, and tactics, techniques and procedures. Recent advances in computing power as well as budget reductions have made the use of modeling and simulation (M&S) even more important and widespread.

Currently, there are major procurement programs underway in each of the branches of the armed forces and in the Department of Defense (DoD) to develop new combat models and simulations. These models seek to improve upon the deficiencies of those in use today, some of which have been in existence for over 15 years. Without exception, each of these programs have had to start from scratch and have reused very few of the components from legacy models. This is mostly due to poor documentation in the models and the fact that many of the models, such as Janus, VIC, and TACWAR, are implemented in procedural languages.

One of the features of object-oriented programming (OOP) is the prospect of reusing design and code on future projects. Reuse is the ability to take a segment of code or an object and use it in multiple programs. However, reuse does not simply happen, it must be planned by thinking beyond the immediate application and developing a more general design. Developing an object model is often the initial step in developing an object-oriented program. An object model is a portrayal of the objects in a system and their relationships; it serves two purposes: as a blueprint for developers and as a communication tool between the customer and the developer.

The Army Modeling and Simulations Office (AMSO) Master Plan is the Army's implementation of the Defense Modeling and Simulations Office (DMSO) Master Plan. It details the Army's modeling and simulation vision, objectives and the standards development process. [AMSO, 1995] Two of the themes that resonate throughout the plan are interoperability and reuse of models and simulations. Interoperability is the ability of a simulation to interact with other models. As a major user of models and simulations, the DoD desires reuse and interoperability to reduce costs and speed up the development of new software. M&S interoperability and reuse can not be achieved without a common technical framework. This common technical framework is the fundamental objective shared by the Army and DoD Master Plans. There are six major objectives in both Master Plans and the Army has developed 18 standards categories that support these objectives as shown in Table 1.

| AMSO Objectives | Standards Categories |
|---|---|
| 1. Provide a common M&S technical framework | • Architecture<br>• Data<br>• Visualization<br>• Object Management<br>• Functional Description of the Battlespace |
| 2. Provide timely and authoritative environmental representations | • Terrain<br>• Dynamic Environments |
| 3. Provide authoritative representations of systems | • Acquire<br>• Attrit<br>• Move<br>• Logistics<br>• C3 Systems<br>• Mobilization<br>• Deployment<br>• Cost Representation |
| 4. Provide authoritative representations of human behavior | • Reasoning<br>• CGF |
| 5. Provide an M&S infrastructure to meet developer and end-user needs | • VV&A |
| 6. Share M&S benefits | |

**Table 1. Standards Categories [AMSO, 1996]**

Several initiatives support this common technical framework. The Functional Description of the Battle Space (FDB) is a simulation-independent data repository that is under development by the United States Army Simulation, Training, and Instrumentation Command (STRICOM). The High Level Architecture (HLA) is a DoD sponsored initiative which primarily seeks to facilitate interoperability among simulations and to promote reuse. Object models are a key feature of both the FDB and HLA. Standard algorithms is a topic heading that includes the development of standard algorithms for battle field functions including, but not limited to, attrition, target acquisition and movement.

This thesis will conduct an analysis of selected simulations to recommend a standard object model with high level abstract classes for use in future object-oriented, entity level combat models and simulations.

## A. BACKGROUND

The following sections are broad overviews of the different subjects that are covered in the thesis.

### 1. Current Army Combat Modeling Initiatives

This thesis is part of an AMSO directed study being performed by the Training and Doctrine Command Analysis Center-Monterey (TRAC-MTRY), in support of the Object Management Standards Category. Object Management is defined as follows:

> The process that develops an Army wide environment where modeling and simulation efforts produce programming objects that are consistent in their representation of object attributes, generally understood by all decision makers and the modeling community, and where they promote simulations with policy compliant objects that are interoperable at all levels allowed by their battle environment. The goals are to use standard object classes and object class attribute definitions in building all future simulations; to develop all future object class code to be HLA complaint; and to develop HLA simulation object model (SOM) policies which will focus on consistent attributes for publication/reflection. [AMSO, 1997]

The object model that is developed in this thesis will be reviewed by the Standards Category Coordination Committee for use in the standard. There are several other standard categories with ongoing work that will impact this study.

***a. FDB***

The development of a simulation system or model requires basic information that describes the physical environment, systems and material contained in the environment, human characteristics, organization, doctrine, processes and their interactions. In the past, the contractors for a particular simulation would be responsible for collecting all of this information. Often the sources of this information were not documented and the data were rarely made available to future developers. Therefore, when a new simulation was developed, there was minimal sharing of information. This led to different developers using different sources for the same information, yielding inconsistent representations of the real world.

To combat this problem, the DMSO Master Plan directed the development of a Conceptual Model of the Mission Space (CMMS). The Functional Description of the Battlespace is a US Army research and development effort managed by the Simulation, Training and Instrumentation Command (STRICOM) in conjunction with the National Simulation Center (NSC). It is the Army's contribution to the CMMS effort. The purpose of the FDB is, "to document the standard descriptions of components and characteristics of battlefield functions that must be represented to produce credible simulations." [Pettitt] The FDB is specifically designed to support the development of the Warfighter's Simulation (WARSIM) 2000 and future simulation efforts in the collection of

validated, standard descriptions of battle field functions, physical algorithms, equipment characteristics and terrain data.

The FBD has the following goals:

- Collect data relevant to the M&S community
- Transform collected data into useful information
- Present information in a domain specific view for each user

Access to non-classified information is via the Internet. The sources for the data are primarily U. S. Army doctrinal publications and activities. [Blakely, 1996]

***b. HLA***

One of the primary uses of simulations in the Department of Defense is training. A key feature of many training simulations is the ability to link users from remote locations together. The architecture that has provided this simulation interoperability is called Distributed Interactive Simulation (DIS); however, there are a number of limitations to DIS. As the number of users increases the speed of the simulation slows down, creating a strict upper limit on the number of units that can participate. Furthermore, DIS only supports interoperability between high resolution or entity level simulations; there is a need to link high and low resolution models together.

In August, 1996, DMSO proposed a successor to DIS called the High Level Architecture (HLA). HLA standardizes the procedures for linking simulations by providing a set of rules for interoperability, a Run Time Interface (RTI), and a format for representing individual simulations and groups of simulations with the Object Model Template (OMT). A group of individual simulations becomes a federation when connected over a network through the Run Time Interface. The OMTs provide object model

depictions of the simulations and the federation called the Simulation Object Model (SOM) and the Federation Object Model (FOM). The SOM consists of an object class structure, an interaction table, an attribute/parameter table and a FOM/SOM Lexicon (data dictionary) for an individual simulation. The FOM includes the same components as the SOM, but is an object model template of the federation. [Larimer, 1997]

The DoD has stated that all simulations must be HLA compliant by fiscal year 2001 or receive a waiver. This primarily applies to simulations that will be linked and it does not apply to closed programs that are used strictly for analysis. [USD, 1996] Since the purpose of this thesis is to develop a standard object model for all types of high resolution simulations, the object model developed must support the HLA standards.

### *c. Standard Algorithms*

The third objective of the AMSO master plan is to, "Provide authoritative representations of systems." The Army supports this objective with 8 standards categories:

- Acquire
- Attrit
- Move
- Logistics
- C3 Systems
- Mobilization
- Deployment
- Cost Representation

With the exception of the last category, all of these categories encompass corresponding algorithms. [AMSO, 1996] Research is underway in all of these categories. As work is completed, the standard object model that is developed in this thesis can be updated to reflect the standard algorithms.

## 2. Object-oriented Programming Overview

Object-oriented Programming refers to a style of programming that develops software as a collection of discrete *objects* that incorporate data structure and behavior. An object consists of the data or attributes that describe the object, together with the methods that operate on this data. Methods have names, may accept parameters as input and may return a value. The combination of a method's name, parameters and output type is called a *signature*. Two or more methods may use the same name but be differentiated by their signatures. This is commonly referred to as *overloading* a method.

A *class* is a set of objects that have common attributes and methods. A class is a template or blueprint for objects and every object is an *instance* of some class. Four generally accepted properties of OOP *are abstraction, encapsulation, inheritance* and *polymorphism*.

*Abstraction* can be defined as the stripping away of irrelevant details in order to concentrate on relevant aspects. It is a means of coping with complexity, and is a natural part of problem solving. Abstraction of a problem leads to a structured class hierarchy where objects with common structure are grouped under a class.

*Encapsulation* is the separation of the internal implementation of an object's data from its public interface, i.e. the parts which are accessible to other objects. This frees the user from needing to know how an object is implemented and allows a programmer to

change the implementation without affecting the user, as long as the interface does not change.

*Inheritance* is the ability of a class to have all of the attributes and methods of another class, possibly adding additional attributes and methods. A subclass extends its parent's or superclass's attributes and methods and adds its own. Classes can be organized into a hierarchy with the most general classes as superclasses and more specialized classes as subclasses.

*Polymorphism* is the ability of objects to present a similar interface but use different implementations. For example, a superclass such as Shape might have a method called draw( ). Two subclasses of Shape could be Circle and Square. Circle and Square would both have a draw( ) method, by virtue of inheritance, but they could override it and implement it as needed.

### 3. Model Resolution

Combat models can be classified as high or low-resolution according to the level of detail present in the model. High-resolution models usually have enough detail to represent individual vehicles and personnel. They are often referred to as entity or platform-level models. These models are typically used in battalion level simulations and below. As the number of units in a simulation increases it becomes more difficult to process all of the required information, so individual platforms are aggregated into units such as platoons and companies. As the size of the simulation increases further small units are again aggregated into larger units; these low-resolution simulations are known as aggregate or unit-level simulations.

The algorithms used to represent combat functions are significantly different in high and low resolution models. Entity level models will usually compute line of sight between individual vehicles and keep track of attrition on an entity basis. As the resolution decreases, attrition is computed on a unit basis through the use of Lanchester equations or some other process.

## 4. Selected Combat Models Overview

As a basis for developing a standard army object model, two legacy (Janus and ModSAF) and two developmental simulations (WARSIM and JWARS) were studied. Janus is a high resolution (battalion level), six-sided closed, stochastic, ground combat simulation used for both training and analysis. Although it is not an object-oriented simulation, an object model of Janus has been developed . [Larimer, 1997] ModSAF (Modular Semi-Automated Forces) is used to populated high resolution training simulations with realistic forces and is one of several SAF programs.

WARSIM is the Army's next-generation simulation to support Force XXI. It will replace the Corps Battle Simulation and the Brigade/Battalion Battle Simulation. It is intended for battalion level training and above. WARSIM is the Army's contribution to JSIMS and it will be HLA compliant. JSIMS is the Joint Simulations System. Though initially focused at the operational level of war for both combat operations and operations other than war, JSIMS will be extensible to the strategic level (e.g., activities involving multiple Commander In Chiefs (CINCs) and/or theaters) and to the tactical level (e.g., the prosecution of individual battles or engagements) to support multi-echelon exercises and contingency planning.

JWARS is a theater-level analytic model, scheduled to be the replacement for TACWAR. JWARS is being developed from an object-oriented model, and is planned to be stochastic, while TACWAR is deterministic. There is a Memorandum of Agreement between JWARS and JSIMS that states that both programs will use the object model that is being developed by JWARS [JWARS, 1995]. The WARSIM program is developing its own object model, so while WARSIM is to be incorporated into JSIMS, it is proposing a different object model.

## B. STATEMENT OF THESIS

A set of standard objects and classes will help maintain consistency among Army models and foster both interoperability and model reuse. (In the above case it would have provided a common starting point for WARSIM and JSIMS, whereas now they are using different object models in their development.) The intent is to give simulation developers a common starting point without dictating implementation.

This thesis will examine the following questions:

- What is the appropriate resolution for standard classes?
- What are examples of appropriate object standards for entity level simulations?
- How are standard object attributes related to standard data and standard algorithms?
- Are there additional standard algorithms or data requirements for the standard objects?

The analysis will consist of the following tasks:

- Examining legacy and developmental simulations to determine the salient features of their class representations.

- Identifying sample standard objects. This will include arranging the objects in a class hierarchy and documenting the rational for the standardization of these objects.
- Demonstrate the ability to cross reference data and algorithm standards with the attributes and behaviors of the standard objects.

The primary data sources are the documentation for the simulations being examined and the data and algorithm standards documents produced by the standards committees.

# II. METHOD

This section discusses the methodology used to create the standard army object model, Rumbaugh's Object Modeling Technique (OMT).

## A. OBJECT MODELING OVERVIEW

A model is an abstract representation of something with the purpose of increased understanding. A model omits nonessential details and is consequently easier to manipulate and understand than the real world entity. Models have been used for centuries and can take many diverse forms, such as blue prints, scale models, and CAD models. Building a model before starting to build a complex system enables the designers to spot flaws. Changing a model is much easier and cheaper than modifying the real thing.

An object is an abstraction with crisp boundaries and meaning for a specific problem. For example a tank, an operations order, and 1st platoon are all objects. An *object model* describes the structure of objects in a system, their identity, attributes and methods and their relationship with other objects. The goal of an object model is to capture the essential concepts from the real world that are important for the problem at hand. The object model serves as a bridge between the customer and the programmers; they are useful both for abstracting a problem and for designing a program.

James Rumbaugh's OMT is one form of object-oriented analysis that produces an object model and it has been used on several DoD programs. This thesis will use a modified form of Rumbaugh's OMT to build the standard army object model. There are three models produced using Rumbaugh's OMT. The *object model* represents the structure of objects in a system - their identity, their relationships to other objects and their

attributes and methods. The *dynamic model* represents those aspects of a system that change over time and the sequence of operations. The *functional model* describes transformations that occur in a model without regard to when those transformations occur. The last two models are primarily concerned with implementation of the model. Since the standard army object model avoids specifying implementation, this thesis will only include the development of an object model.

## B. OBJECT MODEL FEATURES

The object model is represented graphically using object diagrams that contain object classes. Classes each have a unique name and define the attributes and methods carried by each object instance. Attributes and methods can be classified by their level of accessibility. Attributes and methods may be *private, public* or, *protected*. A private attribute or method can only be accessed or changed by the original class in which its is first used. In general attributes should be private to support encapsulation. If it is desirable for other classes to access private attributes, then special methods are provided that allow other classes to change or access this data. These methods are usually called "setter" and "getter" methods, respectively. Public attributes and methods can be accessed by any class. Protected attributes and methods can only be accessed by the original class or its descendants.

The accessibility of all attributes in the standard army object model allow full flexibility in implementation. If the attributes are made private in an implementation, then the appropriate setter and getter methods must be provided. Methods must be made public in order provide a useful interface. Figure 1 details the class notation; the symbol

with the attributes indicates that the implementation decision is left up to the programmer. The symbol associated with the methods indicates that it is public.

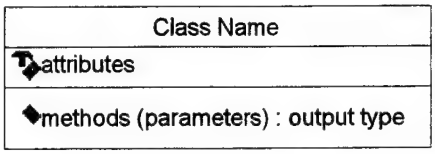


**Figure 1. Class Notation**

Classes sharing common attributes and methods are arranged into *hierarchies*. This defines the subclass/superclass relationship. The superclass is often referred to as the ancestor of its subclasses. Subclasses inherit all of the attributes and methods from their ancestors and can added their own attributes and methods. This relationship is often referred to as an "is-a" relationship. Any descendent is also an instance of its ancestors. Classes can also be *associated* with one or more other classes. An *association* is a conceptual connection between classes. *Multiplicity* specifies how many instances of one class may relate to a single instance of a associated class. Figures 2 and 3 illustrates these concepts.

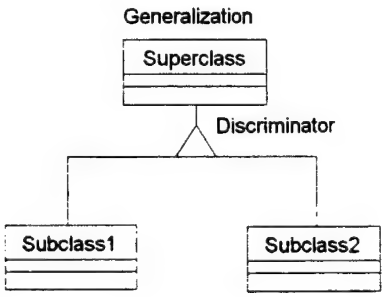


**Figure 2. Class Hierarchy (Generalization)**

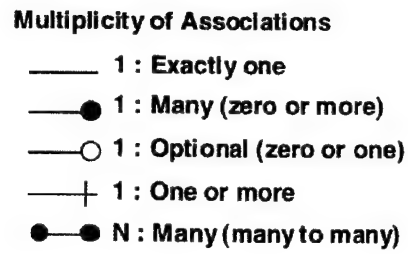


**Figure 3. Associations**

Objects are often made up of other objects. A lamp is made up of a base, a cover, a switch and wiring. In other words a lamp may be viewed as an *aggregation* of its parts. This is often described as a "has-a" relationship, as in the statement, a lamp has a switch. The amount of aggregation in a model is a function of how much detail is desired for a particular problem. Assume that an architect is designing a house using computer aided design software. If the only purpose of a lamp in the design is as a light source, then aggregation may not be necessary; the component parts do not affect the brightness of the lamp. However, if an engineer is designing a lamp, then the aggregation is desirable; aggregation allows the engineer to analyze tradeoffs between different components in the lamp. To illustrate the concepts of inheritance, association and aggregation, the following example is provided:

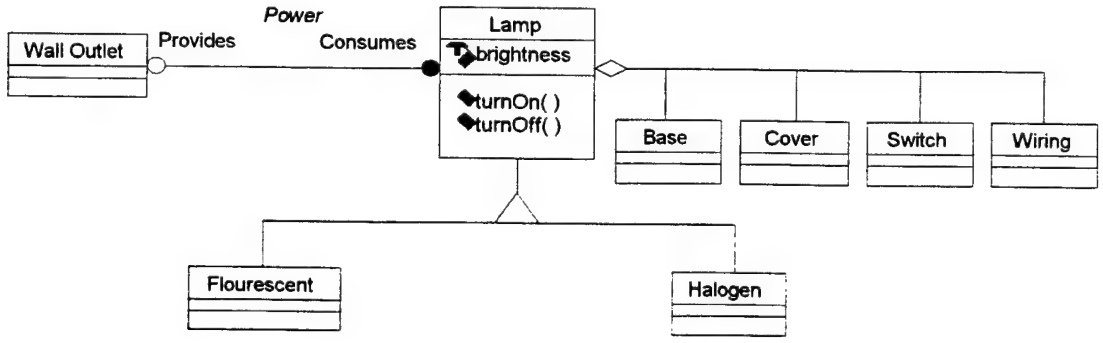


**Figure 4. Object Model Example**

A lamp is made up of (is an aggregation of) a base, a cover, a switch and wiring. Lamps can be classified as Fluorescent or Halogen. Both Fluorescent and Halogen inherit the methods turnOn( ) and turnOff( ) and they both have a brightness. Fluorescent and Halogen are specializations of the Lamp superclass. Lamps are associated with Wall Outlets. Note that this association has a specific name that describes the association. Associations are bi-directional. The name of the association should imply a direction and the inverse relationship is inferred. A wall outlet may provide power for zero or more lamps, while a lamp may draw power from zero or one outlets.

## C. ADVANCED CONCEPTS

The concepts of association, generalization and aggregation will now be discussed in greater detail.

### 1. Aggregation

Aggregation is a strong form of association in which the aggregate object is made up of its component parts. In some cases, making the distinction between aggregation and association is difficult. In general, if two objects are bound by a parts-whole relationship than it should be modeled as an aggregation. If the two objects are usually considered to be independent even though there is a strong link then this should be modeled as an association. The decision to use aggregation or association is often a matter of judgment and there are no firm rules. For example, in Figure 5, a commercial Company is modeled as an aggregation of its Divisions and Departments while Employees are only associated with the company. In the military domain it would be appropriate to model a unit as an aggregation of the individual personnel and platforms in the unit.

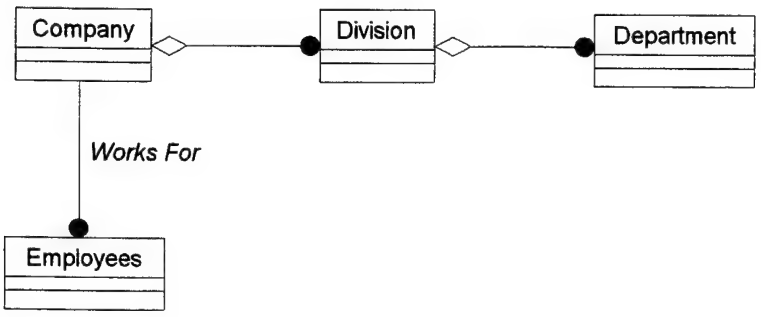


**Figure 5. Aggregation and Association Example**

## 2. Generalization

Aggregation is different from generalization. Aggregation relates instances; two distinct objects are involved. Generalization relates classes and is a way of structuring the description of an object. One of the most important features of inheritance is that an instance of a class is an instance of all of its ancestor classes. This is a guarantee that all descendants of a class will have the methods and attributes of its ancestors. Each instantiated object also knows its class. To illustrate this, let Captain be an ancestor class of PriorEnlistedCaptain (Figure 6).

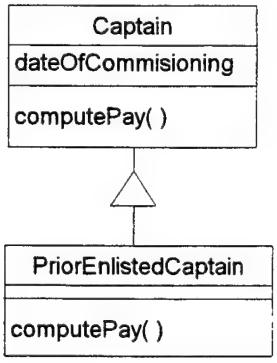


**Figure 6. Inheritance Example**

Captain has an attribute called dateOfCommissioning and a method called computePay( ) where the amount computed depends on the dateOfCommissioning. Note that

PriorEnlistedCaptain is a descendent of Captain. One of the advantages to being a PriorEnlistedCaptain is that the pay is higher than that of a regular captain. PriorEnlistedCaptain would have the same attributes and methods as Captain by virtue of inheritance, but computePay( ) would be overridden to reflect the higher pay status. Suppose that all of the Captains with a certain dateOfCommissioning are told to compute their pay. While all regular Captains with this dateOfCommissioning would compute the same pay, the PriorEnlistedCaptains with this dateOfCommissioning would computed a higher amount.

## D. OBJECT MODEL DEVELOPMENT

The first step in developing an object model is to define the problem. For this thesis, the problem is a general one of modeling land warfare at the platform-level. However, the scope of the problem is not limited to land-based platforms, since air and maritime platforms can both interact with land platforms. Thus any developed standard must support platforms from the air, land and maritime domains.

Rumbaugh provides a list of steps to be followed when developing an object model. The following steps are derived from Rumbaugh's, but they have been modified to reflect the steps followed in the development of the standard army object model:

- Identify objects and classes
- Prepare a data dictionary
- Identify attributes and methods of objects
- Organize and simplify object classes using inheritance
- Iterate and refine the model [Rumbaugh, 1991]

The first step is perhaps one of the more difficult steps in developing a generic object model, since there are no clear cut bounds on the problem, nor is the resolution of the models strictly defined. Should only vehicles be modeled? What about missiles? Should the level of detail include individual bullets? Most of the platform-level models studied limit the level of detail to representing individual vehicles and soldiers. Missiles and rockets are usually modeled, but the level of detail is typically limited to modeling the weapon's effects. A standard object model must be able to accommodate multiple levels of resolution at the entity level.

The purpose of the data dictionary is to clearly define the terms used in the model. Isolated words often have ambiguous meanings that are open to interpretation. Next, attributes and methods are assigned to the objects and the objects are subsequently grouped by common attributes and methods. The classes are then simplified and grouped into an inheritance structure. The final step of refining the model will occur after this thesis has been published when the completed model is reviewed by the Standards Category Coordinating Committee.

## E. STANDARD ARMY OBJECTS

The purpose of the standard army object model for platform-level simulations is to enhance interoperability and reuse. This will be accomplished primarily through the use of the standard army object model as an interface with which objects may interact in a generic way. The standards should impose little restrictions on the manner in which features are implemented. The object model only specifies the minimum number of attributes and methods necessary to perform an object's functions. Additional attributes and methods may be added in any implementation of the object model.

The developed model adopts a component-based approach to modeling platforms. Instead of modeling an entire tank, different components will be combined to model the tank. This gives the modeler much more flexibility and potential for reuse. Flexibility is achieved by the ability to assemble components in different ways. For instance, a tank can be fitted with an alternate main gun or sensor without altering the code that defines that tank. Reuse is achieved by the ability to use the same sensor or weapon on a variety of platforms.

The methods specified in the model provide the means of communication between a platform and its components. They also provide the means of communication between different platforms. This enhances the interoperability between simulations. By providing a standard interface, even if simulations are implemented in different languages, the information that is needed to communicate is already specified.

## F. CHAPTER SUMMARY

This chapter has provided the reader with a basic understanding of the object modeling process. The features and associated notation of Rumbaugh's OMT were introduced and explained. To further clarify key concepts, several examples were presented. The following chapter will analyze existing and future simulations using the OMT notation described in this chapter. The lessons learned from the analysis will then be incorporated into the proposed standard army object model.

# III. ANALYSIS OF LEGACY AND FUTURE MODELS

In this chapter an analysis will be conducted of two legacy simulations (Janus and ModSAF) and two developmental simulations (WARSIM 2000 and JWARS). It will point out the salient features of the models and highlight differences and similarities between the models.

## A. LEGACY SIMULATIONS

Both of the legacy simulations examined are high resolution simulations. They model combat at the battalion level and below and represent most entities individually, although entities may be combined into units.

### 1. Janus

An object model was developed for Janus as part of Larry Larimer's thesis entitled, "Building An Object Model Of A Legacy Simulation." The purpose of his thesis was to develop an HLA Simulation Object Model (SOM) for Janus. Two object models were developed. The first object model was developed by grouping "platforms" according to their physical characteristics. The term platform is used to denote any individual object that can accept orders and performs tasks that are significant in a military context. These platforms were based primarily on the Janus user's data base which lists each platform that the user might introduce to a scenario. Attributes were listed for each platform and platforms with common attributes were combined into classes. The resulting object class hierarchy is shown in Figure 7. [Larimer, 1997]

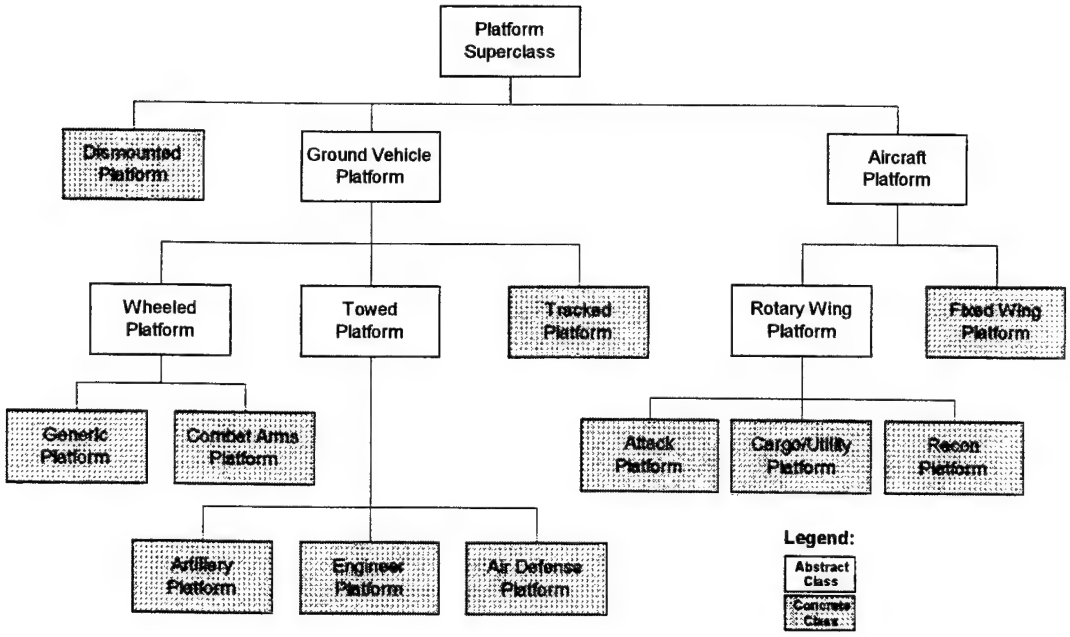


**Figure 7. Janus Platform Class Hierarchy [Larimer, 1997]**

In the second model, the objects were first grouped by Battlefield Operating System (BOS) function, and then further sub-grouped by common attributes. The battlefield operating systems distinguish the role of the platform in combat and include: Intelligence, Command and Control, Maneuver, Mobility/Survivability, Air Defense, Fire Support, and Combat Service Support. This model may be considered more suitable for use by an analyst who may be more interested in a platform's function rather than its physical characteristics and is shown in Figure 34. This second platform class hierarchy illustrates the flexibility of an object model to provide more than one appropriate model of a simulation for military analysis and training. [Larimer, 1997]

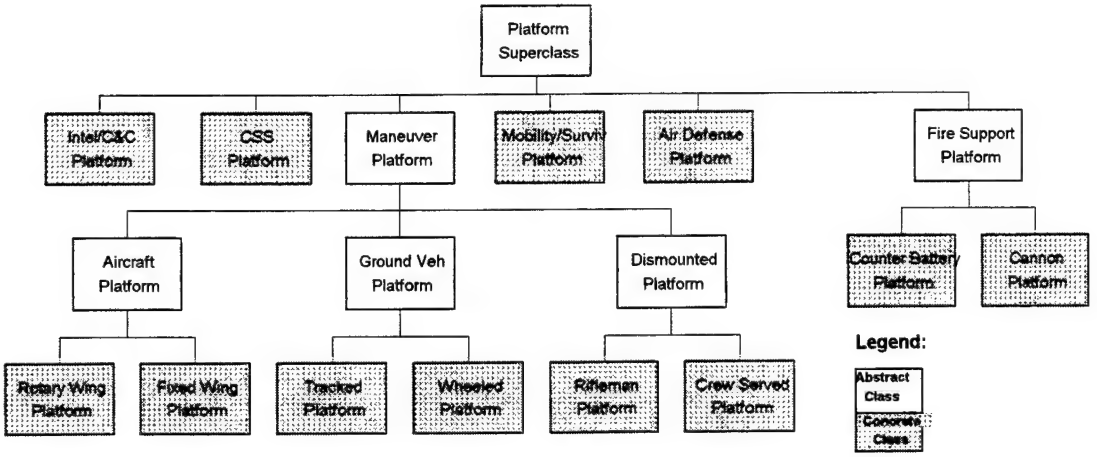


**Figure 8. Refined Janus Platform Class Hierarchy [Larimer, 1997]**

## 2. MODSAF

ModSAF is implemented in Kernigan & Ritchie (K&R) C in order to maximize compatibility with a variety of hardware platforms and because of the run-time efficiency of C. Although K&R C is not an object-oriented language, ModSAF does employ what is termed, "object based programming." Instead of single inheritance, ModSAF uses composition, where larger classes are composed of subclasses. There is a one-to-one correspondence between object classes and certain libraries. [Loral, 1995]

ModSAF is used to generate weapons and vehicles commonly found in land combat, therefore the types of systems represented are limited to ground and air systems and dismounted infantry; maritime units are not represented. There are no published object models for ModSAF. The following object model, presented in Figures 9 - 13, was developed for this thesis based on information contained in the ModSAF user documentation. It is by no means the only possible object model for ModSAF.

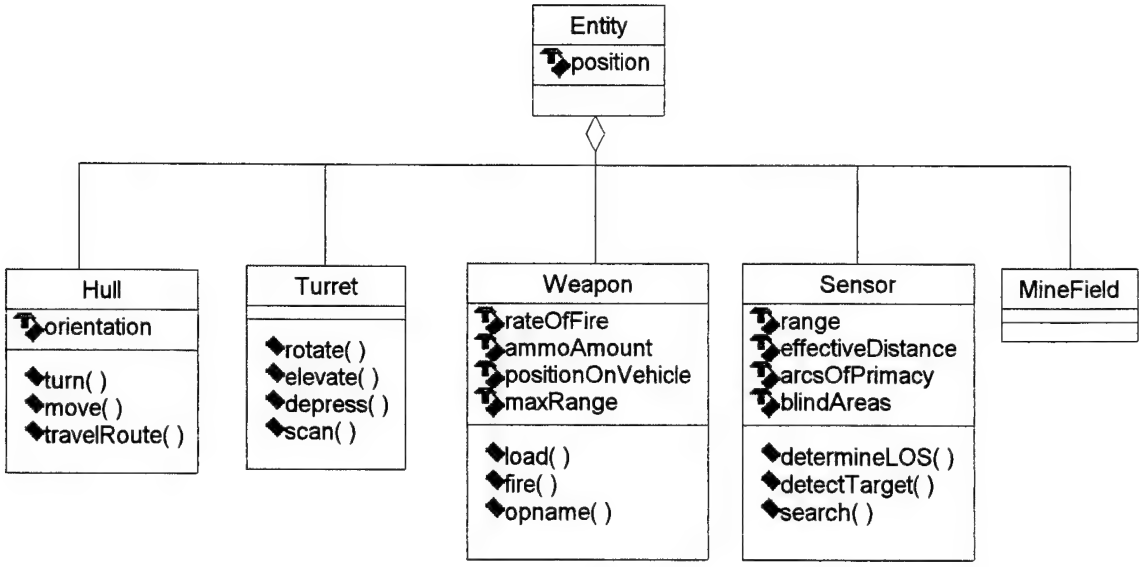


**Figure 9. ModSAF Top Level Entity Object Model**

The ModSAF Entity is equivalent to a Janus Platform. An Entity may have combinations of one or more of the following components: a Hull, a Turret, a Weapon, a Sensor or a Minefield. In order for an object to have a position it must be part of an entity. This means that instead of an instance a Minefield, there is an instance of an Entity that has a Minefield. The Turret class is used to mount weapons and sensors and it provides an Entity with those extra methods that are listed. The Hull, Weapon and Sensor classes are explained below.

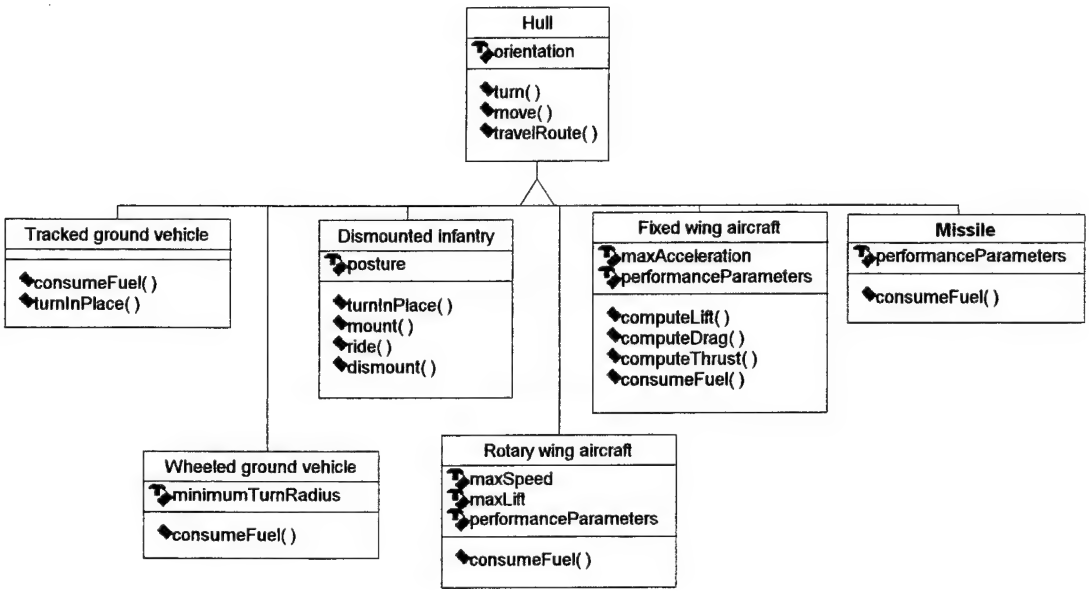


**Figure 10. ModSAF Hull Class**

The Hull class in Figure 10 is used to provide a means of movement for an Entity. The different subclasses of hull provide different types of movement. Missile is a specialize subclass of Hull that has several subclasses as seen in Figure 11. Each of the subclasses of Missile is differentiated by its performance parameters. The Weapon class represents any object that is used to cause damage to an Entity. Finally, the Sensor class gives an Entity the ability to detect other Entities.

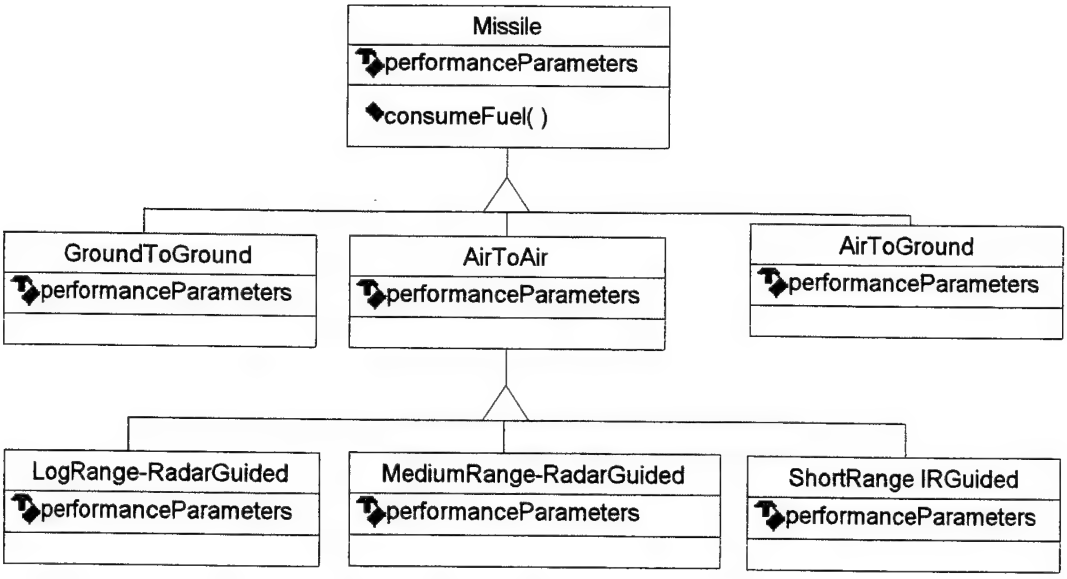


**Figure 11. ModSAF Missile Class**

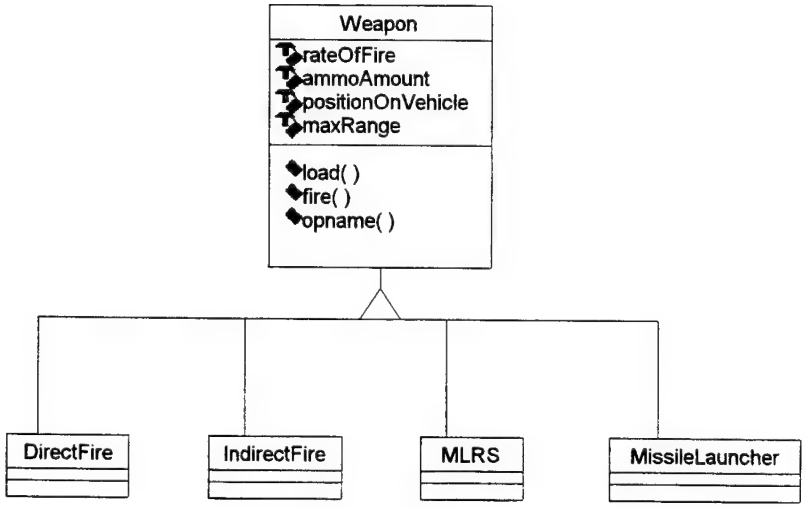


**Figure 12. ModSAF Weapon Class**

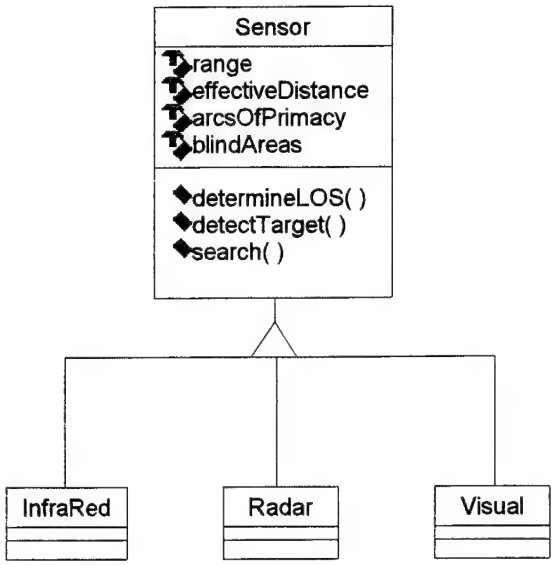


**Figure 13. ModSAF Sensor Class**

ModSAF is used to generate very specific entities for use in distributed simulations. The vehicles and individuals that it can represent are finite and a representative list is included in Table 2. Attributes and methods have been included with classes where appropriate; however, since many attributes and methods are associated with specific vehicles or individuals, they are not be applicable to all of the objects in a class. Some of the methods not included in the object model are supply( ), receive( ) and assess( ). For example, not all Entities, such as one having a Minefield, would need the ability to supply other Entities or to receive supplies.

| United States Entities | Russian Entities | German Entities |
|---|---|---|
| M2 Bradley IFV | T-80 medium battle tank | LEO1A5 Leopard medium battle tank |
| M3 Bradley IFV | T-72 medium battle tank | LEO2 Leopard II medium battle tank |
| M1A2 Abrams main battle tank | BMP1 armored fighting vehicle | MARDER1A3 armored fighting vehicle |
| M106A1 tracked mortar carrier | BMP2 armored fighting veh. | MTW M113 observer veh. |
| M109A1-A6 SP howitzer | Mi-24 HIND attack helo | JAGUAR1 |
| M113 ambulance | MIG-27 Flogger fixed wing aircraft (FWA) | JAGUAR2 |
| M113 observer | MIG-29 Fulcrum FWA | SKORPIAN |
| HMMWV | SU-25 Frogfoot FWA | PAH1 helo |
| M88A1 tank recovery veh. | ZSU23-4 23mm air defense veh. | Dismounted Infantry |
| M977 HEMTT - Cargo | 2S1 | |
| M977 HEMTT - Fuel | 2S6 | |
| Dismounted Infantry | BRDM-2 | |
| A-10 Thunderbolt FWA | BTR-80 | |
| F-14D Tomcat FWA | BTR 60PU artillery veh. | |
| F-16D FWA | URAL 375C combat support veh. | |
| OH-58D Kiowa scout helo | URAL 375F fuel truck | |
| AH-64 Apache attack helo | Dismounted Infantry | |
| Avenger air defense | 2B11 towed mortar | |
| M2 Stinger air defense | 2S19 | |
| M270 MLRS | BM21 | |
| M981 FISTV | Mi28 Rotary wing aircraft (RWA) | |
| M992 | Mi8 RWA | |

**Table 2. ModSAF Entities [ModSAF]**

## B. DEVELOPMANTAL SIMULATIONS

WARSIM and JWARS are simulations at the battalion level and above; however, the object models developed for each simulation have an entity-level resolution. Both simulations followed modified versions of Rumbaugh's OMT in developing their object models.

### 1. WARSIM 2000

An equipment object model was developed for WARSIM 2000 as part of the software requirements analysis phase of the software development. This object model was

developed using a condensed version of Rumbaugh's OMT and was based primarily on the systems requirements document. [Castner, 1996] The steps that were followed are:

- Identify and define object classes
- Identify attributes and operations of object classes
- Identify associations between object classes
- Organize object classes using inheritance [Hopkins, 1997]

Figure 14 depicts the top level view of the equipment object model. Equipment is composed of supplies and simulated physical things. A simulated physical thing (SPT) is that class of objects that, "has attributes of activity and state and is subject to attrition and detection."[Hopkins, 1997] SPT is equivalent to an Entity in ModSAF or a Platform in Janus.

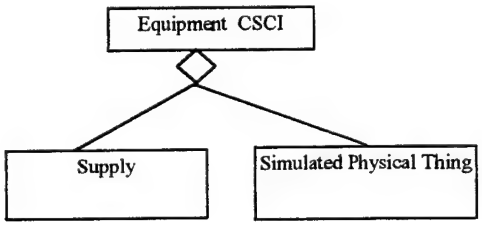


**Figure 14. WARSIM 2000 Top Level Equipment Object Model [Hopkins, 1997]**

The Supply class is further specified as shown in Figure 15. The subclasses shown correspond to standard classes of supply used in normal army operations. Note that Equipment Platform appears as a subclass of supply. Equipment Platform also appears in Figure 16 as a subclass of SPT. This is an instance of multiple inheritance and does not conform to HLA specifications. Since this is a requirements object model, multiple inheritance may not be a problem. There are programming techniques that allow much of

the functionality of multiple inheritance without using multiple inheritance. An example is the use of interfaces in Java.

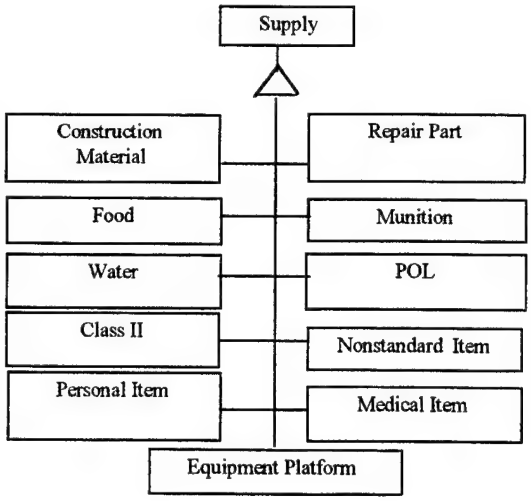


**Figure 15. WARSIM 2000 Supply Class [Hopkins, 1997]**

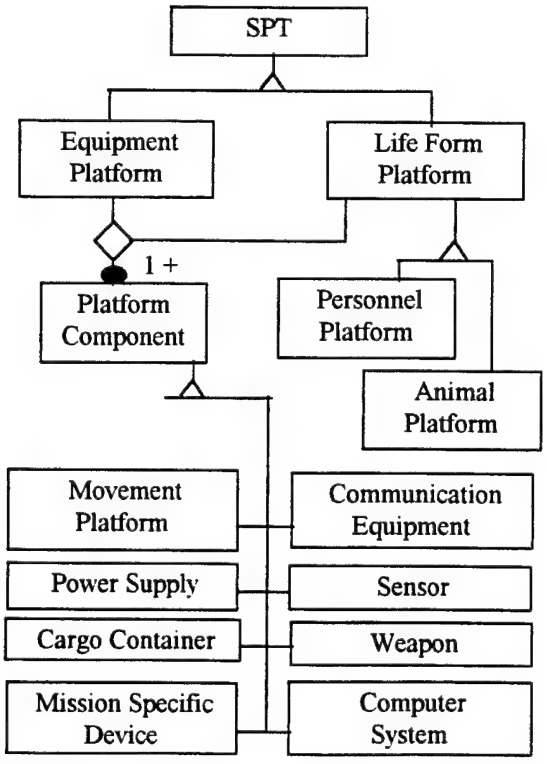


**Figure 16. WARSIM 2000 Simulated Physical Thing Class [Hopkins, 1997]**

SPT is the foundation class for all Equipment and Life Form Platforms. Both Equipment and Life Form Platforms are composed of one or more Platform Components. The Movement Platform class, shown in Figure 17, provides the means of locomotion for all Equipment and Life Form Platforms. The darkened inheritance symbol below Movement Platform denotes the ability to overlap subclass specializations, another form of multiple inheritance. An example is an amphibious vehicle that would exhibit behavior of a Ground Platform and a Water Platform. Apparently, all Life Form Platforms would have a Living Body Platform, although there is nothing in the object model that forbids an Equipment Platform from having a Living Body Platform. The Life Form Platform class may be redundant; none of the attributes or behaviors that might differentiate a Life Form Platform from an Equipment Platform are listed.

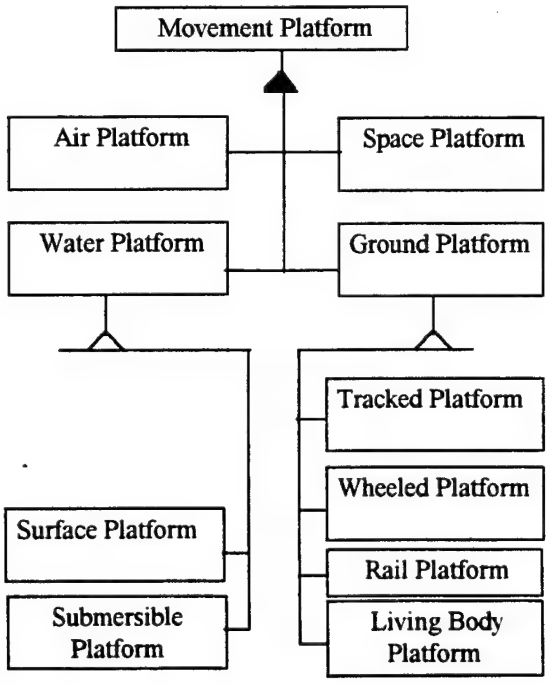


**Figure 17. WARSIM 2000 Movement Platform Class [Hopkins, 1997]**

Weapons are composed of exactly one Aiming System and exactly one Delivery System, as shown in Figure 18. Again, note the use of multiple inheritance implied by the darkened inheritance symbol. The subclasses of Weapon are very similar to the ones used in the Missile and Weapon classes in ModSAF. Although there is not a one-to-one correspondence between classes in the two models, there are definite associations. ModSAF's generic concepts of Direct and Indirect Fire Weapon classes in Figure 12 are included as specific *Surface* Direct and Indirect Weapon classes in WARSIM 2000, while Air to Air and Air to Ground *Missiles* in ModSAF, Figure 11, are replaced by WARSIM 2000's more generic Air to Air and Air to Ground *Weapon* classes. The Hand to Hand Weapon class is unique; WARSIM 2000 is the only simulation included in this thesis that models hand-to-hand weapons.

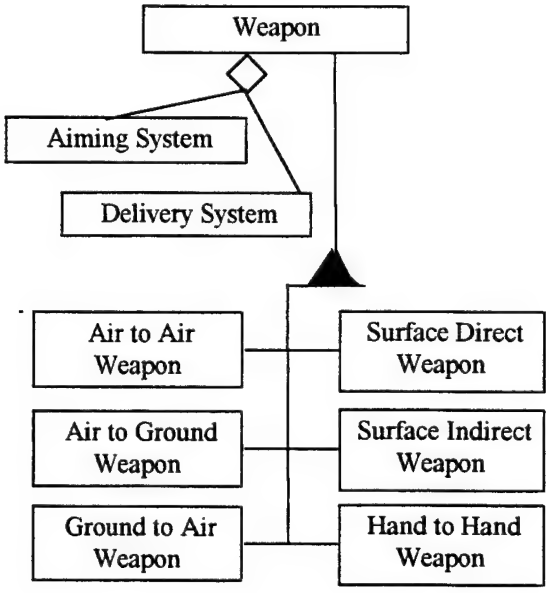


**Figure 18. WARSIM 2000 Weapon Class [Hopkins, 1997]**

WARSIM 2000 is required to allow commanders to interact with the simulation while using organic communications equipment. The Communication Equipment and Computer System classes, seen in Figures 19 and 20, seem to have been tailored to meet this requirement. Communication Equipment can be composed of one or zero transmitters and one or zero receivers. Communication Equipment is associated with a Signal class that is specialized into Text, Data, Voice and Image classes. All of the Signal subclasses are types of data that are transmitted on organic communications equipment.

The Computer System class is unique to WARSIM 2000 and is tailored to meet the requirement of communicating with organic communications and computer equipment. For example, the MC Ghost ABCS, Army Battle Command System, class is used to represent live ABCS equipment used by the training audience. This would allow the live equipment to be detected and to interact with simulated equipment. [Hopkins, 1997]

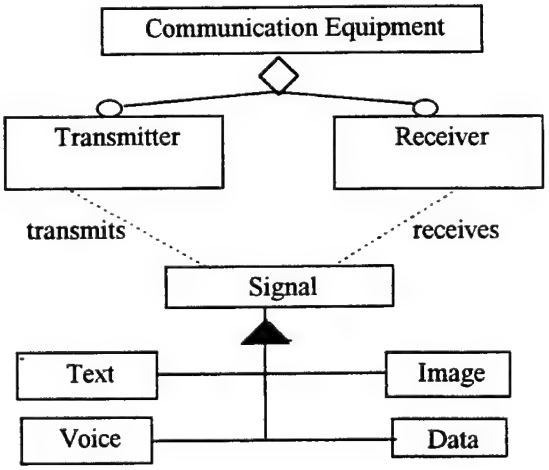


**Figure 19. WARSIM 2000 Communication Equipment Class [Hopkins, 1997]**

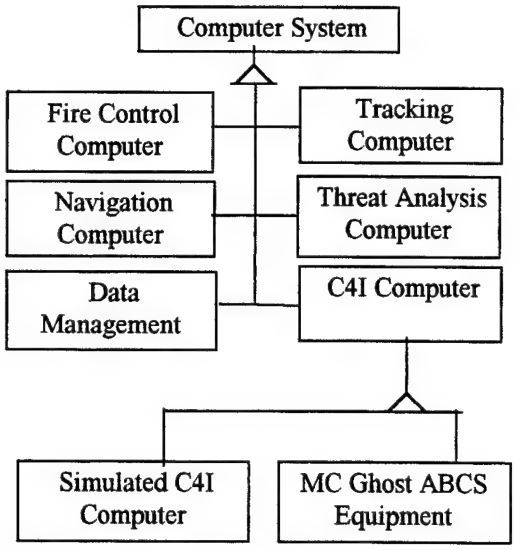


**Figure 20. WARSIM 2000 Computer Equipment Class [Hopkins, 1997]**

Sensors are treated in a straight forward manner and the sensor class is shown in Figure 21. The final subclass of SPT is Mission Specific Device. This class is not elaborated upon at all and may be a catch-all class for objects that do not fall into any of the previous categories.

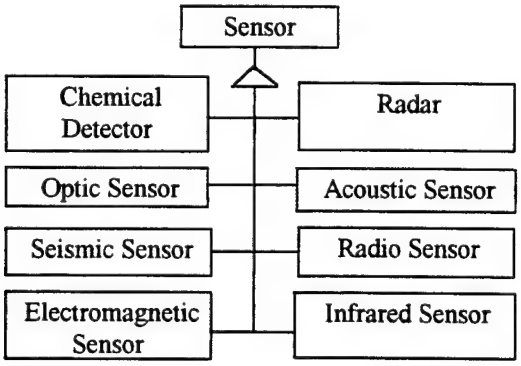


**Figure 21. WARSIM 2000 Sensor Class**

## 2. JWARS

The object model developed for JWARS is based on a Joint Mission Space Model (JMSM) and the JWARS Testbed Scenario and Use Cases. JMSM is the end product of researching, assembling and cataloging the knowledge necessary to define the entities, tasks, and interactions that will be represented in JWARS. [JWARS, 1996a]

The scope of JWARS covers a vast spectrum of requirements. Given the complexity of the JWARS mission space, the program office decided to focus analysis on specific functionality during the simulation development. The testbed scenario is the first of several scenarios that are planned to be used guide JWARS development. Additional scenarios are designed to increase the functionality of the model. Each scenario is a specific application of JWARS as an analytic tool.

The object model that was developed as a result of this first testbed scenario is very large, consisting of over seventy diagrams that detail the object, dynamic and functional models. This thesis will analyze only those object model diagrams that describe the entities modeled in JWARS.

The top level object model for JWARS is shown in Figure 22. This diagram represents the major object classes identified in the analysis of the JWARS problem domain. The border shading surrounding the classes means that additional diagrams exist defining the object class in subsequent sections of The Joint Warfare System Object Model. [JWARS, 1996b]

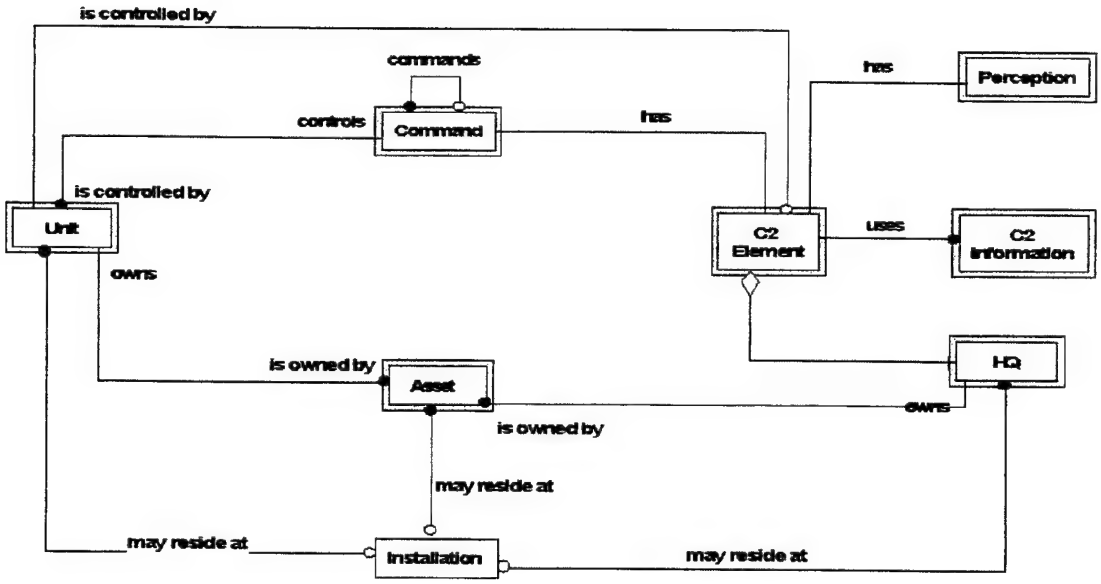


**Figure 22. JWARS Top Level Object Model [JWARS, 1996b]**

Assets denote any item used by a unit to perform its mission. This is the class that is of primary interest and is detailed in Figure 23. Assets (other than Composite Assets) are used to represent all primitive types of useful and valuable things. Composite Assets are used to represent all types of Assets made up of other "primitive" Assets. A Composite Asset can be composed of one or more "primitive" or Component Assets (e.g. a specific Aircraft maybe composed of Sensors, Weapons, etc. and may carry Personnel and Supplies). Note that a Composite Asset in JWARS is equivalent to a SPT in WARSIM 2000, an Entity in ModSAF and a Platform in Janus. Since Composite Asset is also a subclass of Asset, it inherits all of Asset's attributes and methods. This relationship also means that Composite Assets can be composed of one or more other Composite Assets, as would be they case of an aircraft carrier with an air wing embarked. Since Composite Asset is the only class with methods to move, receive attack and assess attrition, most battlefield entities will be an instance of a Composite Asset.

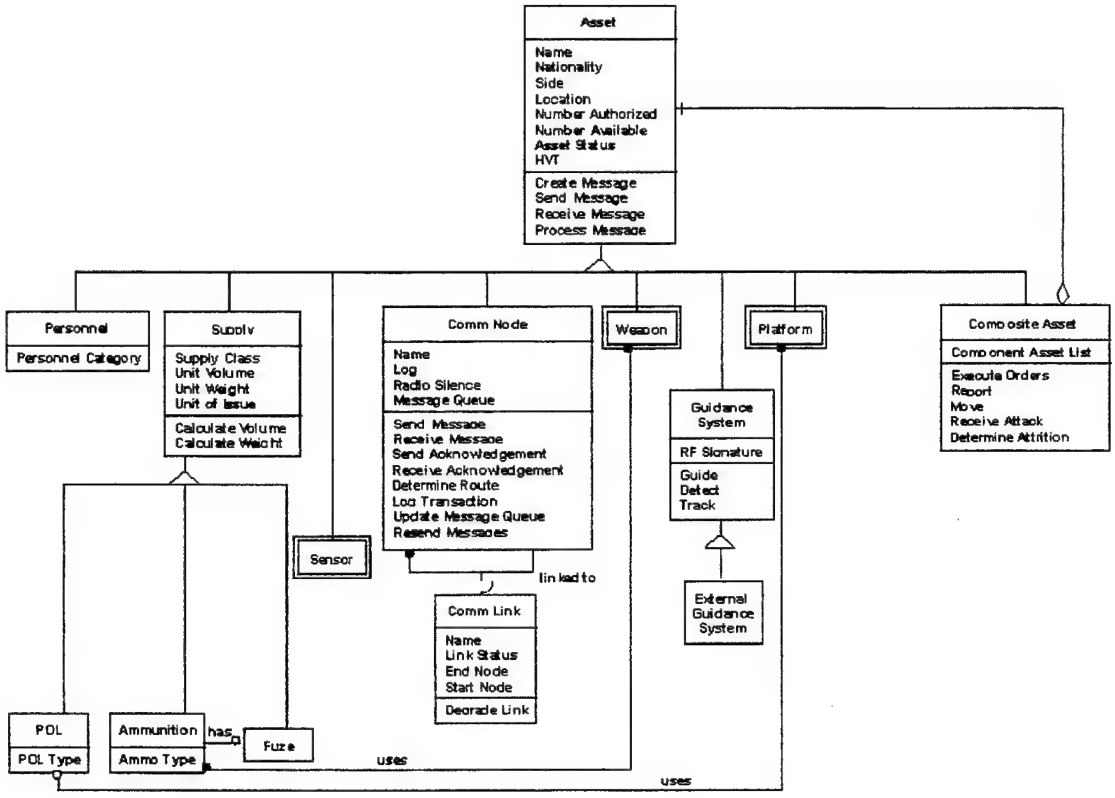


**Figure 23. JWARS Asset Class [JWARS, 1996b]**

The Personnel class can be used to create dismounted infantry, as part of a Composite Asset or to act as the crew for a Composite Asset. The Supply class can exist separately from Composite Assets; however, a Composite Asset is never considered a subclass of Supply as SPTs were in WARSIM 2000.

A Comm Node is a type of Asset that can be accessed via another Asset, a C2 Element, or a HQ. These classes have the ability to create, send, receive, and process messages via the Comm Link. A Comm Node is physically located at the HQ or as part of Composite Asset, thus it can be destroyed or degraded. The Comm Node represents the logical interface to the remainder of the military communications network.

The Sensor Class shown in Figure 24 is an abstract class derived from the Asset class. It is used to build any type of sensor system installed on any type of Asset or Composite Asset. There are two basic categories of sensors, active and passive. Radar's are active sensors. The JSTARS Radar is an example of a composite asset with a Multi-Mode Radar. Multi-Mode Radar is an aggregation of the MTI and SAR classes and can operate in either mode (MTI/SAR).

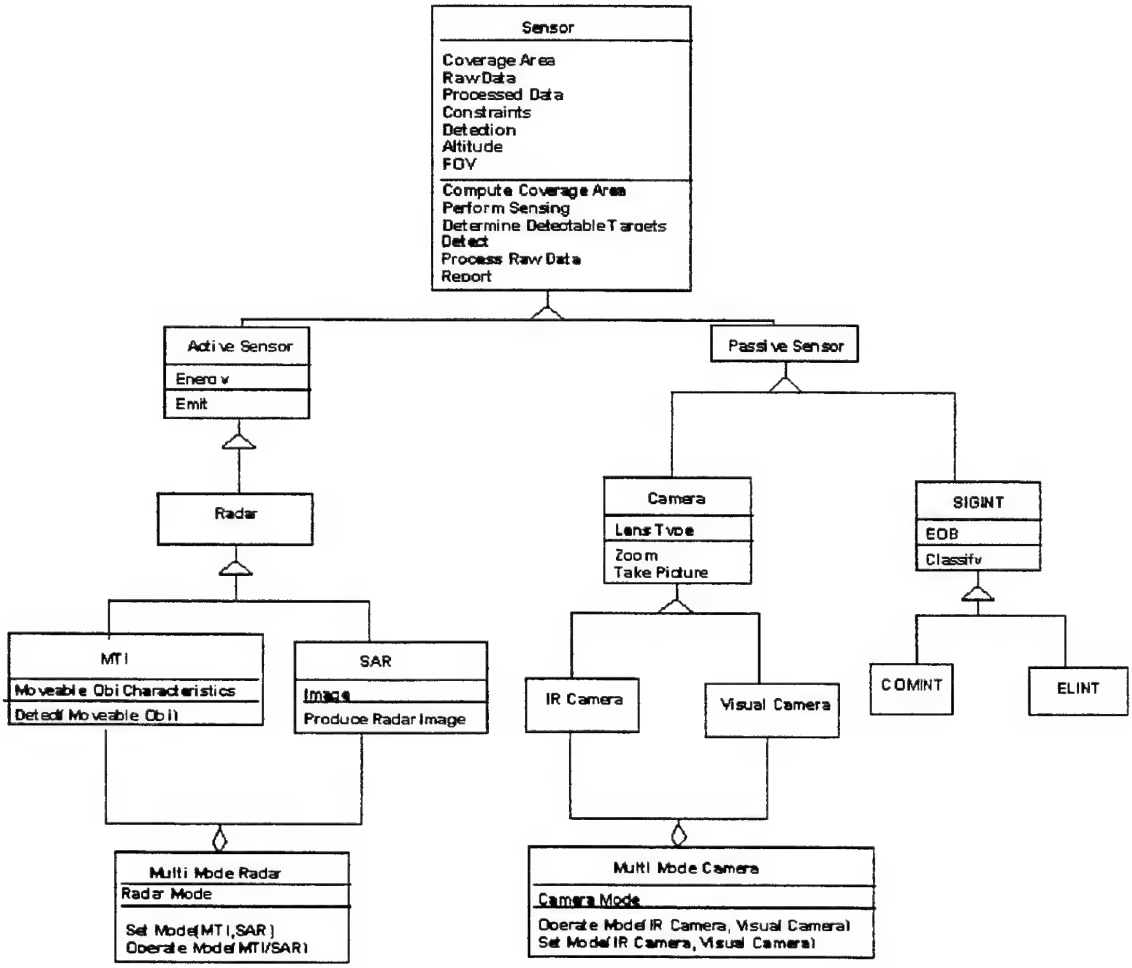


**Figure 24. JWARS Sensor Class [JWARS, 1996b]**

Camera and SIGINT are Passive Sensors. Cameras can be IR or Visual. The Multi-Mode Camera is also an aggregation, combining characteristics from the IR and Visual Cameras.

The SIGINT class is a specialized Passive Sensor that is used to detect electro-magnetic radiation. [JWARS, 1996b]

The Weapon class, as shown in Figure 25, represents any instrument or device used in the attack or defense in a fight or combat. Lethal and Non-Lethal Weapon are two direct subclasses of Weapon. Weapon, Lethal Weapon and Non-Lethal Weapon are all abstract classes. Lethal Weapon represents any class of weapons which are designed to cause casualties or produce deadly effects to Composite Assets. The Non-Lethal Weapon class represents any weapon which is not designed to induce casualties directly. [JWARS, 1996b]

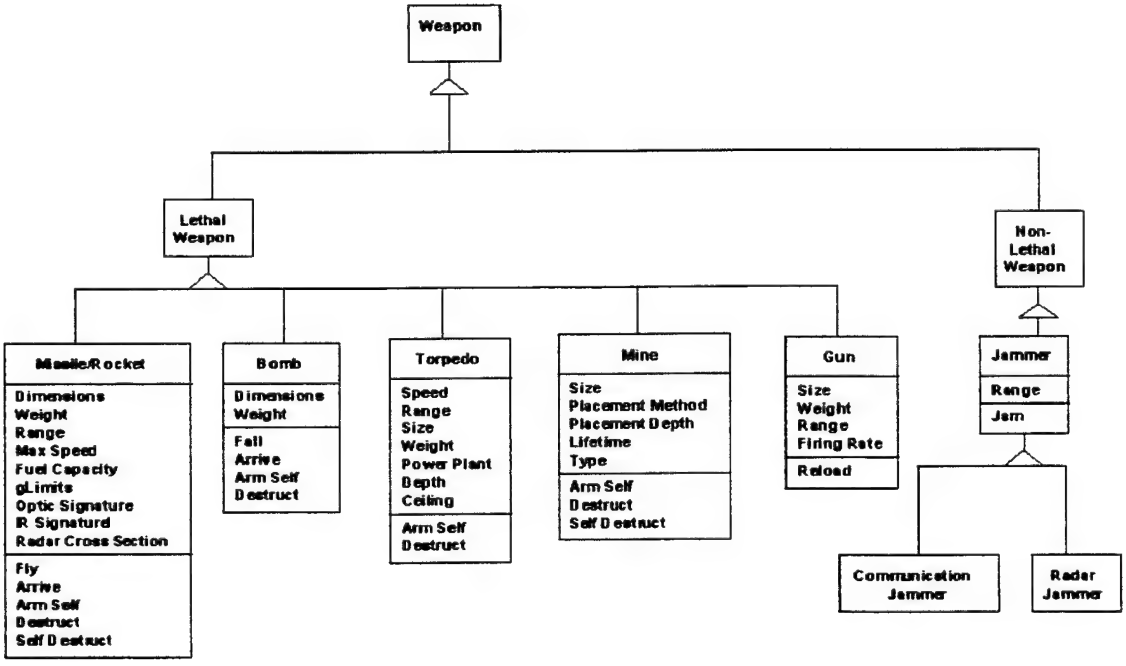


**Figure 25. JWARS Weapon Class [JWARS, 1996b]**

The Missile/Rocket class represents weapons that are thrown, shot, or propelled at a target. The Bomb class represents weapons that are filled with explosive material and detonated by a Fuse. The Torpedo class represents various subsurface explosive devices

for destroying ships or submarines. The mine class represents various floating or moored explosive devices used to blow up ships, or on land against personnel and vehicles. The Gun class represents any device for shooting ammunition. The Jammer class represents any device that causes interference with detection. Instances of Missile, Rocket, Bomb, and Torpedo classes can have Guidance Systems attached to them as part of a Composite Asset. The Guidance System can be internally or externally controlled. [JWARS, 1996b]

The Guidance System in JWARS is equivalent to an Aiming System in WARSIM 2000. Beyond that there appears to be very few similarities between the Weapon class in JWARS and those in ModSAF and WARSIM 2000. This highlights the variety of ways in which weapons systems can be categorized. Some of the difference in JWARS may stem from the limitations of the testbed scenario being used to develop the current object model. Expanding the Gun class into Direct Fire and Indirect Fire classes is an example of how one class might be expanded.

The Platform class is used to represent transportation means associated with Units, C2 Elements, and Composite Assets and is shown in Figure 26. Platforms are currently of type Land Vehicle, Aircraft, Maritime Platform and Space Platform. The Aircraft class represents any machine supported for flight in the air by the dynamic action of air on its surfaces. The Aircraft class is the only Platform class that is expanded. This is a limitation imposed by the testbed scenario which places a heavy emphasis on air platforms. The Land, Maritime and Space Platform classes represents a self-propelled, boosted, or towed conveyance for transporting burdens on land, sea or in space, respectively. [JWARS, 1996b]

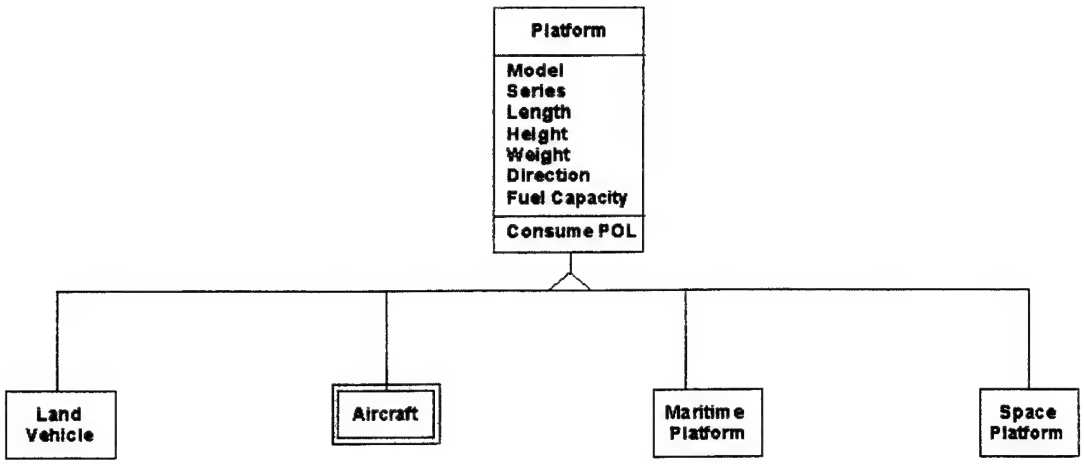


**Figure 26. JWARS Platform Class [JWARS, 1996b]**

JWARS Platform class in equivalent to ModSAF's Hull class and WARSIM 2000's Movement Platform class. Unfortunately, the term "platform" is used in WARSIM 2000 to also designate Equipment Platforms and Life Form Platforms. There is a one-to-one correspondence between the subclasses of JWARS Platform class and WARSIM 2000's Movement Platform subclasses (Figure 17). WARSIM 2000 adds more detail to its ground and Water Platform classes, in contrast to JWARS expansion of its Aircraft class, which is shown in Figure 27.

The Aircraft class is characterized by various performance factors such as maximum speed, cruise speed, maximum combat radius, etc. Fixed Wing and Rotary Wing are two direct subclasses of Aircraft. The Fixed Wing class represents air platforms which have wings that are rigidly and permanently attached to the fuselage. Rotary Wing class represents air platforms which have one or more airfoils that rotate about an approximately vertical axis. Both Fixed and Rotary Wing classes have specialized subclasses: Fighting Aircraft, Combat Support Aircraft and Transport Aircraft. [JWARS, 1996b] The Fighting and Combat Support Aircraft class are shown in Figures 28 and 29.

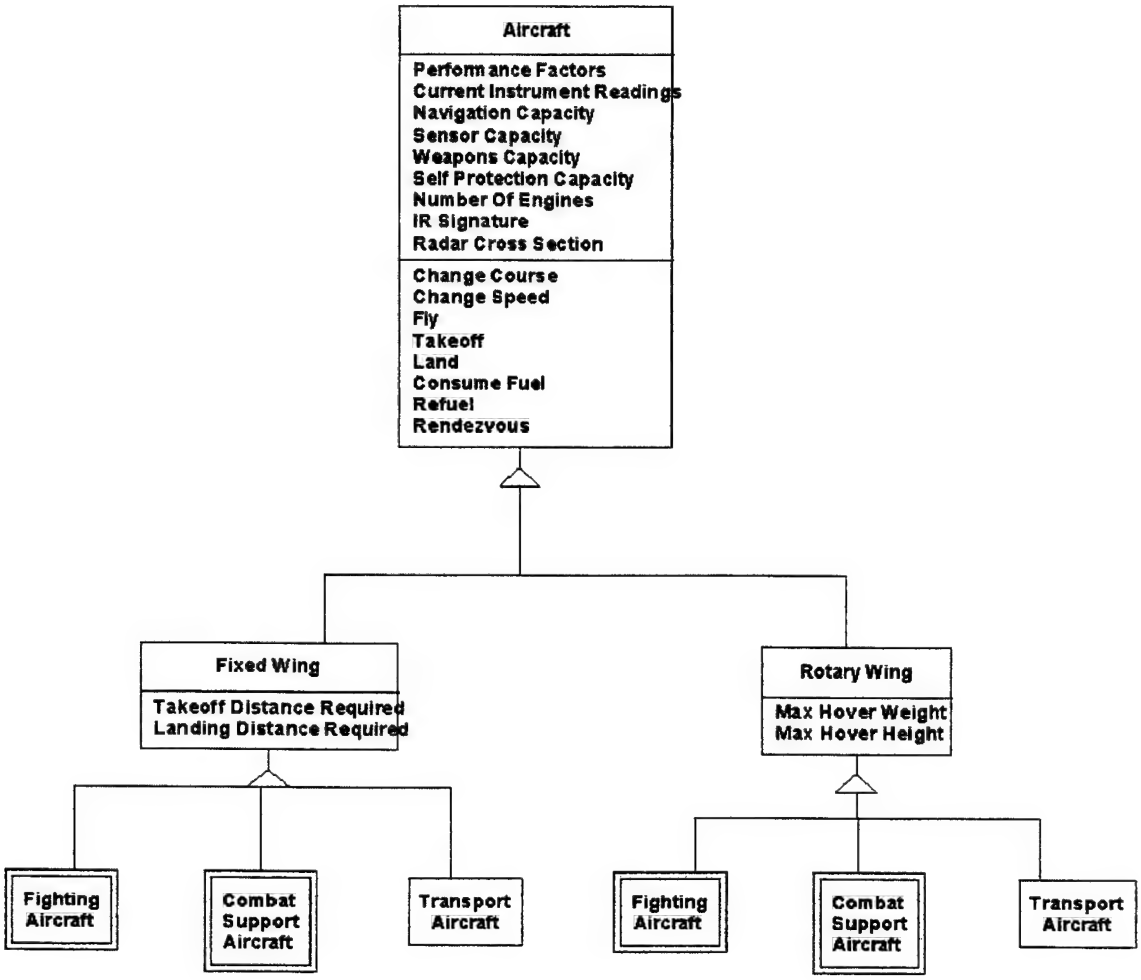


**Figure 27. JWARS Aircraft Class [JWARS, 1996b]**

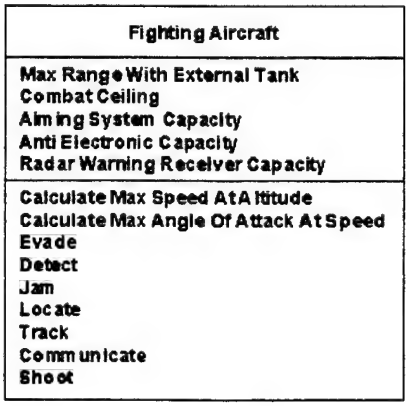


**Figure 28. JWARS Fighting Aircraft Class [JWARS, 1996b]**

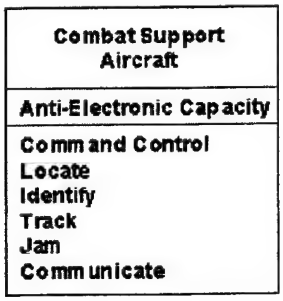


**Figure 29. JWARS Combat Support Aircraft [JWARS, 1996b]**

## C. DISCUSSION

Three of the models analyzed, ModSAF, WARSIM 2000 and JWARS, use a modular approach in developing their object models. The object model developed for Janus is a classification that was based on the common attributes and methods of the platforms present. This approach was used to develop the HLA SOM; however, a modular approach is also possible since the Janus platform has weapons, sensors and other components.

There are similarities between all of the object models and any of the object models can adequately represent the common army entities found on a modern battlefield, e.g. tanks, infantry fighting vehicles, dismounted infantry, aircraft, etc. The biggest differences between the object models are in the nomenclature used to represent nearly identical objects. The primary example of this is that all four simulations use a different term for battlefield entities. At a minimum, a standard army object model will provide a common language for the development of new simulations.

# IV. STANDARD PLATFORM-LEVEL ARMY OBJECT MODEL

The role of the standard object model is to enhance interoperability and to achieve a minimum level of uniformity in Army ground combat simulations. The standard object model will detail a set of classes with a minimum level of attributes and methods. The standard classes are not comprehensively detailed. This gives as much flexibility as possible to simulation developers.

## A. HIERARCHY MODEL

Rumbaugh's OMT provides a series of steps that are followed when developing an object model. The first step of Rumbaugh's OMT is to identify the objects and classes in the problem domain. If an object model is being created for a well-defined simulation, then identifying objects is not a problem. Although the number of objects might be large, there are clear guidelines for inclusion of objects in the model. This is not the case for developing a standard object model because there is not a well-defined problem statement or even model resolution.

Even for a well-defined problem, arranging the objects into a single class hierarchy for a combat simulation is difficult. The same set of objects can be classified several ways. Usually features of interest are placed high up in the class hierarchy. This is illustrated in Figure 27 of the JWARS object model, where Aircraft are first classified by flight characteristics and then by primary mission. This scheme could easily be reversed; Fighting Aircraft, Combat Support Aircraft and Transport Aircraft could be ancestor classes of the Fixed Wing and Rotary Wing classes as shown in Figure 30.

A similar challenge occurs when arranging ground platforms into a hierarchy. Often the grouping that is chosen is a function of the purpose of the model; this was the case with the HLA object models developed for Janus. Since the standard army object model is independent of implementation, an alternate methodology to classification was employed.

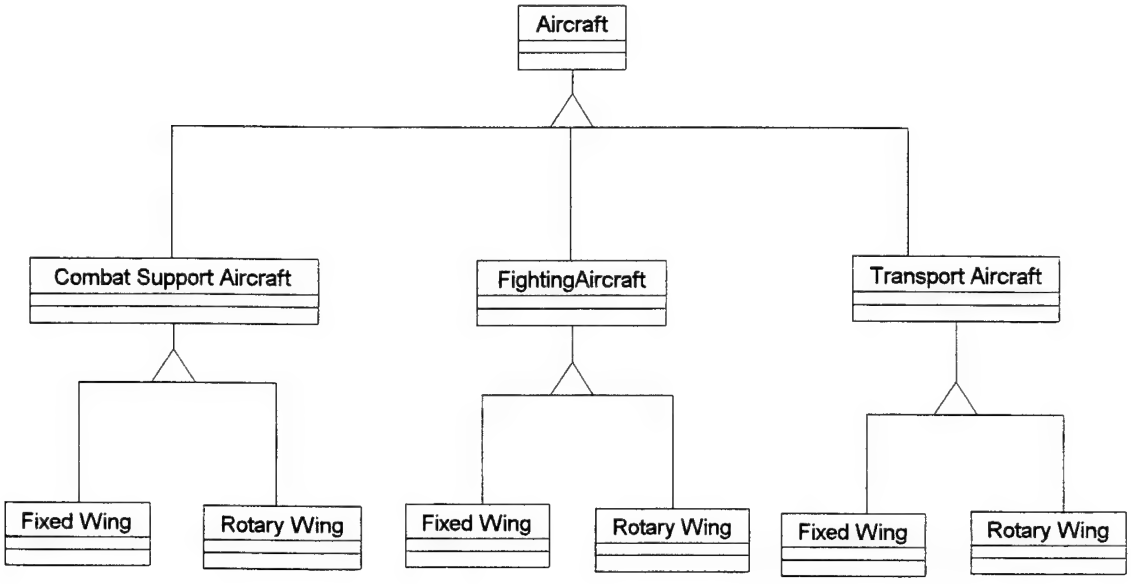


**Figure 30. Alternate JWARS Aircraft Class**

## B. COMPONENT BASED MODELS

One of the most versatile entities in the Army is the HMMWV. Depending on how it is configured the HMMWV can be an anti-tank platform, anti-personnel platform, an observation platform, a command and control platform or even an ambulance. This wide variety of behaviors is achieved by adding and removing the appropriate components. Adding a TOW missile system makes the HMMWV an effective tank killer. Swap the TOW missile system with a laser designator/range finder and the same HMMWV becomes a forward observation vehicle. If a similar approach is taken to

modeling battlefield entities, then the task of developing a standard object model is greatly simplified.

The same modular approach was used to develop the standard army object model. The model is documented with a streamlined form of Rumbaugh's OMT notation. Classes are identified and organized for each of the component superclasses. As the classes are introduced they are defined, and similarities to other models are emphasized. Attributes and methods are included with the classes. Each method may have arguments and return a value; however, these parameters are not specified in the model. Selection of a (standard) algorithm will dictate these parameters. This provides simulation developers with as much flexibility as possible.

### 1. Platform and Platform Components Classes

The fundamental object in the model is the Platform. A Platform is defined as something material that may be perceived by the senses and has the ability to carry weapons or perform militarily useful functions. A unit does not meet this definition because a unit may not be perceived by the senses. A group of platforms may be perceived by the senses, but additional information is needed to classify a group as a unit. A Platform Component is anything that may be added to or mounted on a Platform to extend the Platform's capabilities. The standard Platform Class and its associated Platform Components are shown in Figure 31. The Platform class is also shown in Figure 32, which emphasizes the aggregation.

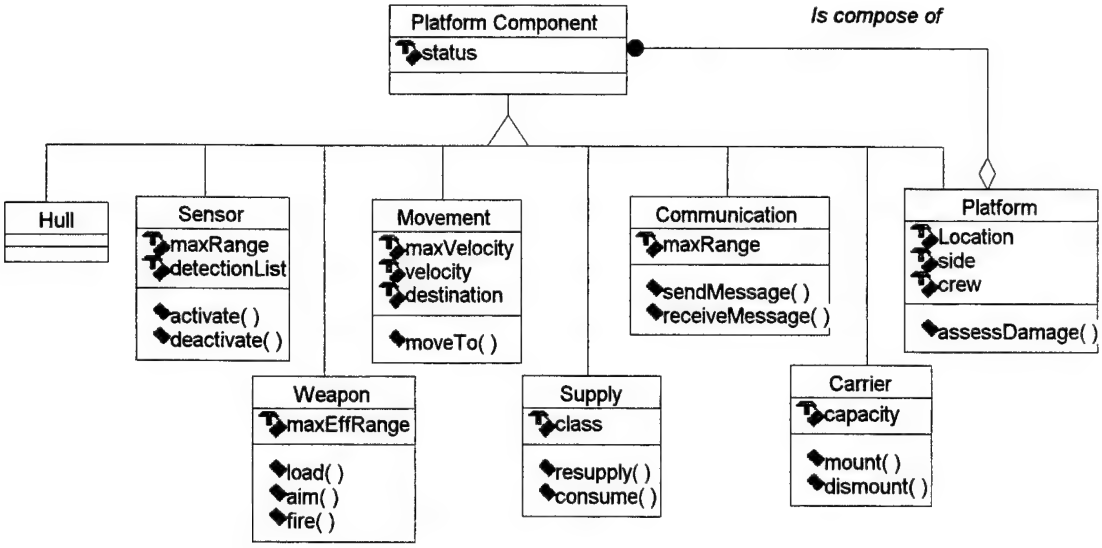


**Figure 31. Standard Platform and Platform Component Class**

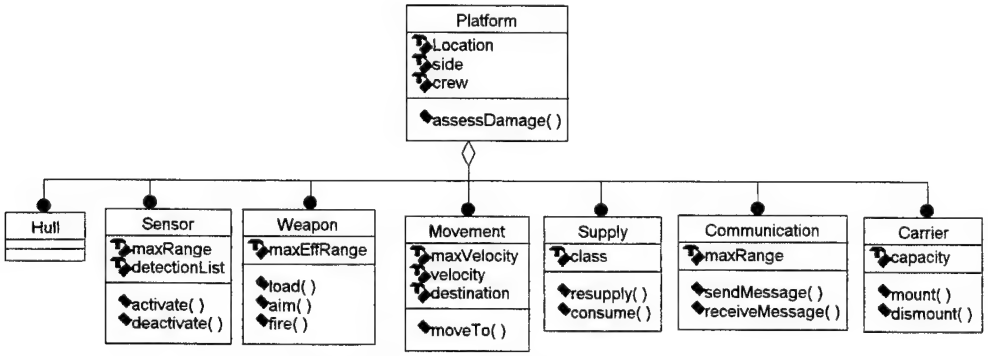


**Figure 32. Platform Class Aggregation**

The Platform Component Superclass has a single attribute of status. All descendants, including the Platform class, inherit this attribute. The status attribute is meant to indicate the degree of functionality of a Platform Component or a Platform. The simplest implementation would be a boolean variable that would indicate alive or dead. Other implementations could indicate a percent of capability or use a multi-dimensional variable.

The Platform class has attributes of Location, side and crew. Platform is the only class that has a Location and a method to assessDamage( ). This dictates that all entities in the simulation that are subject to attrition and have a location must be a descendant of Platform. The assessDamage( ) method is used to update the status of the Platform's Components. The Platform Class is equivalent to the Platform class in Janus, the Entity class in ModSAF, the SPT class in WARSIM 2000 and the Entity class in JWARS.

The side attribute is used to differentiate between Platforms on different sides in a battle. The crew attribute is used to keep track of how many crewmembers are parts of a Platform. The crewmembers do not need to be explicitly modeled and this parameter may simply be set to zero; however, since Platform is a Platform Component, the crewmembers of a Platform may be modeled as other Platforms. This would allow the crew to have mobility and to change vehicles if their original one is damaged or destroyed. This design feature may be more useful in training simulations as opposed to analytic simulations.

Location is also a class and is shown in Figure 33. Every combat simulation uses some concept of Location. The subclasses of Location represent the two fundamental ways to represent location. The Local Class represents those coordinate systems based on Mercator Projection. The UTM grid systems used in military maps and for survey is an example of this type of system. Geocentric coordinate systems are based on a spherical surface that represents the Earth. UTM coordinates are usually used for land based simulations while latitude and longitude are most commonly found in naval simulations. The convert( ) and distanceFrom( ) methods could be overloaded to allow use with either coordinate type. The purpose of the Location class is to allow simulations to internally

represent location with the method that is best suited for its problem domain, while also enabling a certain degree of interoperability.

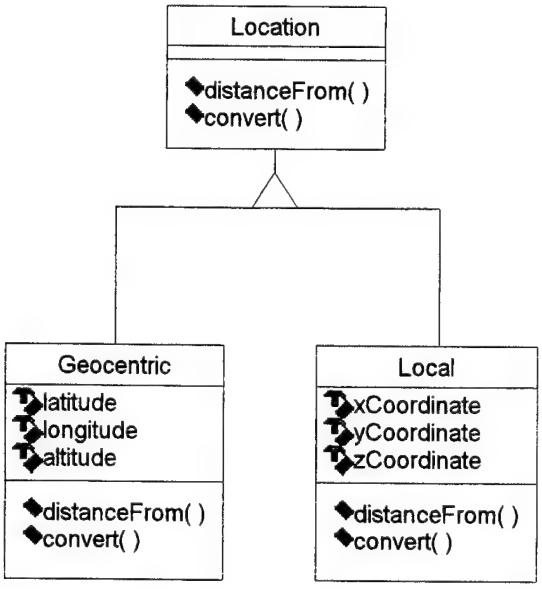


**Figure 33. Standard Location Class**

### 2. Communication, Supply and Carrier Classes

The Communication class is used to represent the ability for Units and Platforms to communicate with each other. The use of the Communication class allows this ability to be explicitly modeled (and thus possibly degraded). Simulations that do not explicitly model communications would not use this class. The Communication class is equivalent to the Communication Node class in JWARS and the Communication Equipment class in WARSIM 2000.

The Supply class is intended to represent things that are consumed by Platforms. The Supply class enables the chosen logistical constraints to be modeled. The class attribute corresponds to the standard supply classes used in the DoD. The resupply( ) and consume( ) methods would be used to increase or decrease the number of supplies. Some

aspects of supplies being consumed could easily be incorporated into methods of other classes. For instance the fire( ) method in the Weapon class could make a call to the consume( ) method of the appropriate Supply subclass to account for expended ammunition. There are Supply classes in both JWARS and WARSIM 2000. Janus and ModSAF keep track of certain supplies such as fuel and ammunition.

The Carrier class is intended to model the capability of certain Platforms to carry other Platforms, such as an IFV carrying a squad of infantry or an ammo carrier carrying ammunition. The capacity attribute places a physical constraint on the amount that can be carried. The Cargo Container class in WARSIM 2000 may be equivalent; however, the only associations outlined are between the Cargo Container class and the Supply class. [Hopkins, 1997]. JWARS does not explicitly model a Carrier class, although a Composite Asset may have one or more Assets or Composite Assets (Figure 23). Certain Platforms can mount and dismount on other Platforms in Janus.

**3. Weapon, Sensor, Hull and Movement Classes**

The Weapon class is used to represent all objects that are designed to cause damage to another object. Guns, bullets, bombs and shells are considered to be weapons. The concept of weapons is represented in all of the simulations included in the thesis and is usually central to any combat simulation; however, there is some disparity between the studied simulations as to what exactly constitutes a "weapon." For instance, JWARS and WARSIM 2000 both consider missiles to be weapons while ModSAF considers a missile to be a specialized Hull Class. Modeling a missile as a Weapon may cause some difficulties, as some simulations may explicitly model the flight of missiles, which would require a Location to be associated with the missile. One possible solution to this dilemma

is to model a missile as a Platform but have it implement an interface with the Weapons Class.

The Movement class is used to represent the means of propulsion for a platform. The Movement class is abstract and it is refined into four subclasses that represent fundamentally different means of movement: Maritime, Ground, Space and Air. These classes are further refined in Figure 34. The listing of the moveTo( ) method in each of the subclasses indicates that this method is overridden.

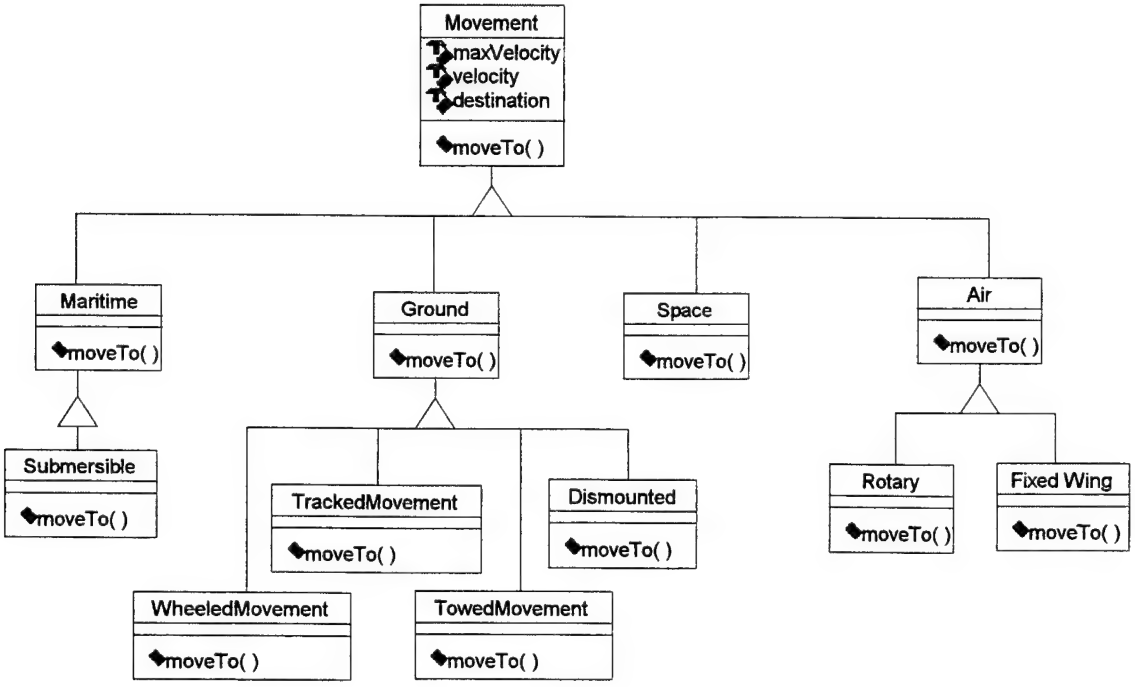


**Figure 34. Standard Movement Class**

The Hull class is designed to hold the physical characteristics of a particular platform. Since most simulations model different physical characteristics, it was desirable to separate these attributes from the platform class. Although there is a strong connection between the Hull and Movement classes (an armored hull usually implies tracked movement) it is beneficial to separate the two ideas. This allows the modeler to change

the physical characteristics of a Platform without necessarily changing any movement algorithms. The Hull class is the logical place to hold any attributes that would be required by the Sensor class such as IR signatures or cross-sectional area. The Hull class could also be used to model armor characteristics. If this level of detail is not modeled then the Hull class is not needed. Features of the standard Hull class are included in JWARS's Platform class and ModSAF's Hull Class. Due to the variety of ways to model physical characteristics, no specific attributes are listed.

The Sensor class is intended to contain attributes and methods associated with all sensors and is shown in Figure 35. A Sensor is defined as any device that responds to physical stimulus and is used to establish the existence or location of an entity. Sensor classes are included in ModSAF, WARSIM 2000 and JWARS, but the criteria used to subclass the Sensors differ. The standard Sensor class has Active and Passive sensors as its two major subclasses. Active sensors rely on returns from transmitted energy while passive sensors do not transmit energy. The maxRange attribute describes the physical limitations of the Sensor. The detectionList attribute is meant to keep a list of all detected entities. The activate( ) and deactivate( ) methods allow the Sensor to be turned on and off. The details of the sensing algorithm used are hidden in the activate method.

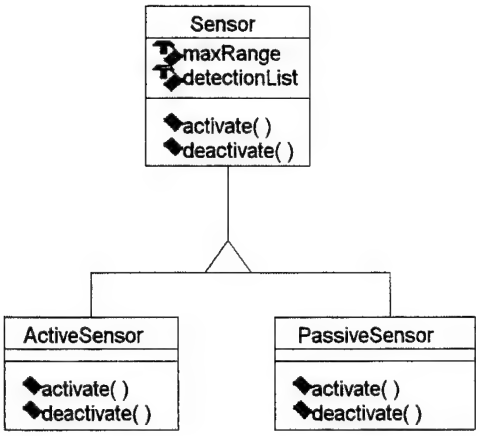


**Figure 35. Standard Sensor Class**

## 4. Platform Example

The two following examples demonstrate how the abstract classes presented so far can be used to create concrete platforms. The first example specifies the minimum classes that are required to create a generic tank. This is the most amount of detail that the standards should specify. The second example extends this tank into an M1A2, a specific tank model.

### *a. Generic Tank Example*

An example of how a platform may be subclassed into a specific type of vehicle is shown in Figure 36. The Tank class is a descendant of the Platform class. The Tank class has exactly one Hull, and one Tracked Movement component. There are one or more Active and Passive Sensors contained in the Tank, such as laser range finders and IR sights on the main gun. A Tank is modeled with at least one DirectFire Weapon, its main gun and possibly auxiliary DirectFire Weapons. Since it can fire projectiles with its main gun there is at least one supply class. More could be added for additional weapons or to keep track of fuel levels. Finally there is at least one Communication class, which

allows the Tank to communicate with other objects. There are no additional attributes or methods added to this class, only the types and numbers of components are specified.

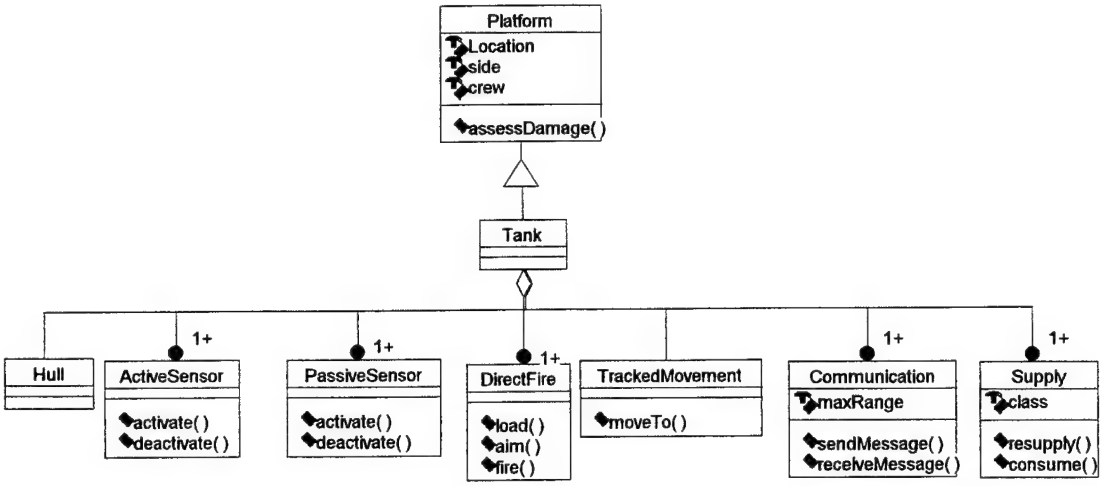


**Figure 36. Tank Class**

### *b. M1A2 Example*

Figure 37 shows an example of an M1A2 tank object that might be used in a hypothetical simulation. This is an extension of the Tank class specified in Figure 36. The parent classes are shown in parentheses. The cardinality of each of the Platform Components is one, except for the VHFRadio, which has a cardinality of two. More specialized classes have been added to the model and additional attributes are added to some classes based on details in the simulation. For instance, the Tank(Hull) class has additional attributes (IRSig, and visSig) based on the types of sensor that are used in the simulation. The MainGun(Weapon) and CoAxMG(Weapon) classes have orientations that are independent of the Tank(Hull) class orientation. The activate( ) and deactivate( ) methods are overridden for each of the Sensor subclasses and the moveTo( ) method is overridden for the TrackedMovement class.

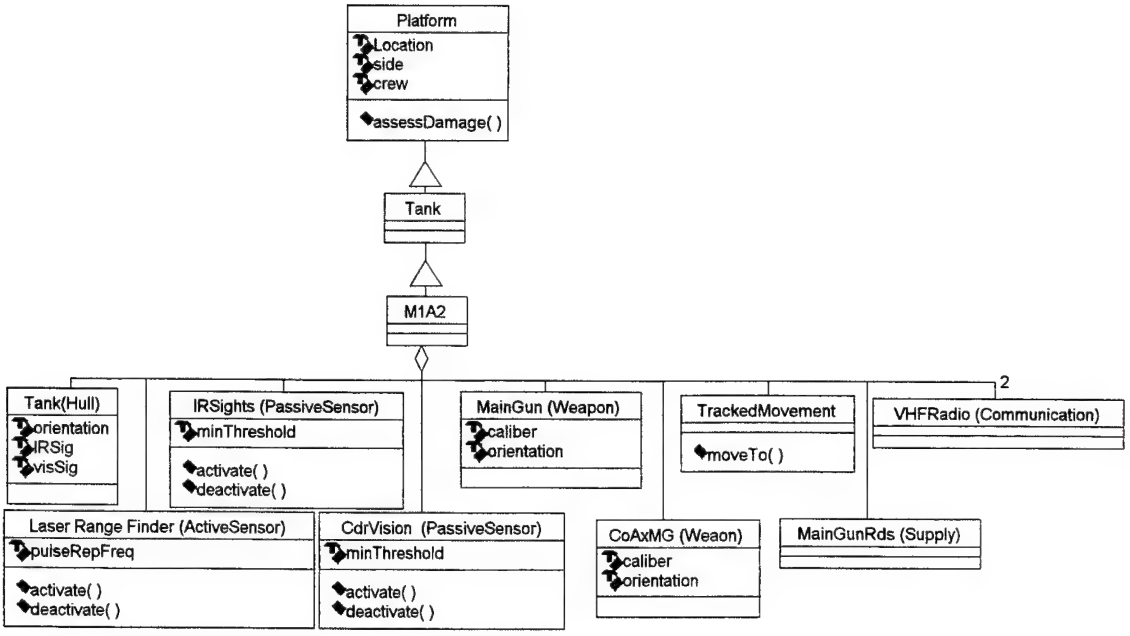


**Figure 37. M1A2 Class**

## C. STANDARD ALGORITHMS

The establishment of a collection of standard algorithms has three major benefits:

- Enforcing model consistency
- Supporting verification and validation
- Supporting simulation development

However, standards do have drawbacks. First, there is the possibility of adopting an inadequate or even incorrect algorithm as the standard. Second, standards eliminate the ability to cross check results derived from different approaches and assumptions. Finally, standards may stifle innovation; a developer with a better algorithm may abandon it if there is already an accepted standard. [AMSAA, 1996]

The standard army object model is capable of adapting to a wide variety of algorithms. This allows new algorithms to be evaluated without creating a new

simulation. The following example illustrates how an attrition algorithms can be incorporated into the Standard Army Object Model.

There are two basic steps in most high-resolution attrition algorithms. First an algorithm determines if a target was hit. If there was a hit then the target assesses the damage. A proposed standard algorithm requires the cross sectional area of the target that is visible in order to compute an aimpoint. This establishes the need to add a determineCrossSection( ) method to the hull class. The method is incorporated into the Hull class because cross section is a function of physical attributes. This method would need the Location of the Weapon that is aiming, thus Location would be an argument to the method. Finally the target needs have an orientation, so this attribute is also added to the Hull class. The modified hull class is shown in Figure 38.

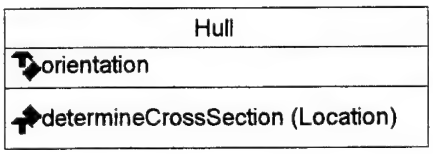


**Figure 38. Modified Hull Class**

After an aimpoint is computed by the aim( ) method, the fire( ) method would determine if there was a hit based on the cross-section that was visible and the range to the target. If there was a hit then the target Platform's assessDamage( ) method is invoked. In order to compute the amount of damage, the target must know the type of weapon that was used, the range of the shooter and the angle of impact. Thus the assessDamage( ) algorithm requires a shooterLocation which would have type Location and a weaponType which would be a Weapon subclass. The resulting method's signature looks like this: assessDamage(shooterLocation: Location; weaponType: Weapon). The algorithm would

update the status of all of the components contained in the Platform based on the amount of damage. [AMSAA, 1996]

By adopting this standard algorithm, the need for adding specific Weapon and Hull subclasses was made apparent. The algorithm used to calculate damage was based on a table that accepted type of weapon and the type of hull as two of its arguments. Some of the hull types included in the database are tanks, armored infantry fighting vehicles, light armored vehicles, SP howitzers, and trucks. Some of these classes are refined even further. Tanks are sub-classified by whether or not their ammunition is compartmentalized. The weapon types are even more numerous. While all of these types can be included in the standard object model, to do so now would dictate implementation to developers.

## D. DISCUSSION

The components that were modeled were based on basic battle field functions: shooting, movement, detection, communication and control, and sustainment. This is a logical division and its is shared by the object models that were analyzed.

Currently, none of the associations between the classes are included in the model, although some are obviously implied. This is because implementation can be dictated by associations and the standard army object model is designed to be independent of implementation. The lack of associations also limits the signatures that are used with the methods. This hampers interoperability; if the signatures for a method are NOT specified then developers might as well use a different name for the method than the one that is specified. In the above example some signatures were specified based on a proposed

standard algorithm. Thereafter, any simulation that used the same attrition algorithm would know the signature and the information required for the assessDamage( ) method.

Standard algorithms and data are still under development; once they have been completed, some associations may be outlined in the standard object model. One of the dangers of developing standard algorithms independently is that different groups will most likely come up with different classification schemes for the same objects. The attrition category committee developed a variety of hull types for use in attrition algorithms. If the acquisition category committee does not use the same hull types for its sensor algorithms, then there is a fundamental conflict. The standard army object model is an excellent tool to display the interaction between the recommended standards and to resolve potential conflicts.

As standards are approved and incorporated into the standard army object model, additional attributes will be added and signatures will be defined. This will increase the amount of overhead that the standard will impose on developers. For instance, if a sensing algorithm that uses IR signatures is adopted as a standard algorithm, then all simulations will be required to add this attribute to the Hull class.

The FDB is still under development and currently contains limited numbers of weapons and platform. Much of the data that is required by the standard object model is contained in the existing version of the FDB. All of the vehicles listed had a maximum velocity listed and most of the weapons had a maximum effective range, although there were some notable exceptions, such as the main gun on the M1A2 main battle tank. The standard object model should serve as a link between simulation developers and the FDB

developers, to ensure that the data used in the standard algorithms is contained in the FDB.

The component classes that were developed are designed to be generic enough that most simulations should be able to map their functionality into the Platform Component structure. Therefore, the standard object model can also serve as a bridge between legacy and developmental simulation.

# V. CONCLUSION AND RECCOMENDATIONS

## A. SUMMARY

This section contains conclusions and recommendations for the use of the Standard Army Object Model. This thesis has developed a prototypical standard army object model. Early work attempted to classify different types of platforms and equipment. This proved to be too limited and unwieldy. Further analysis yielded a model that uses a basic Platform class to which any number of standard components can be added. This capability is key to providing for simulations that are flexible and extensible.

All of the simulations studied had very similar object models whose functionality was contained by the standard army object model. The only class that could not be cleanly mapped into the proposed standard was the Computer class in WARSIM2000. However, since this class is designed to meet a specific need for WARSIM2000, it should not be needed in other simulations and therefore it was not included in the standard. However, it is noted that there are no restrictions on adding additional classes beyond those in the standard.

## B. INTEROPERABILITY

The standard methods contained in the standard army object model allow the communication between a Platform and its components. Placing these methods as high up in the class hierarchy as possible maximizes the benefits of polymorphism and allows other objects to access these methods without needing to know specific class of an object. Specification of this minimal set of methods provides a ready interface for use with other simulations.

## C. REUSE

Polymorphism also allows the substitution of compatible components in a Platform. For example, a tank may be fitted with an improved main gun or sensor without having to alter the code that defines the tank or its other components. This provides the opportunity to create libraries of standard components for use with multiple simulations. This is a very important feature for use with analysis of alternatives when procuring new weapons systems. It also allows simulations to grow as new systems are added to the Army's inventory. One of the reasons that so many new simulations are developed is that older ones are unable to easily model new equipment.

## D. FOCAL POINT

As standard algorithms are developed, they may be incorporated into the standard army object model. This will further define the model and highlight inconsistencies between the algorithms, e. g. two separate committees using different Hull types. The FDB is also under development. It is essential that the FDB contains the data required by the standard algorithms. The standard army object model can be used to state these data requirements and can also help to organize the collected data using its components class hierarchies.

## E. AREAS FOR FURTHER STUDY

This is the initial draft of the standard army object model. It will be reviewed by the Object Management Standards Category Coordinating Committee in October 1997. Several of the standard algorithm committees also have products due by the end of the calendar year. The object model proposed in this thesis should be updated to reflect any recommended changes and the adopted algorithms.

Modeling and software development are iterative processes. As the development of WARSIM 2000 and JWARS progress, there will most likely be changes to their object models. This may suggest additional refinements for the standard army object model as well. Finally, if the standard army object model is successfully adopted, then standard object models should be developed for the other services as well as a joint model.

## LIST OF REFERENCES


Army Modeling and Simulation Office, *The Army Modeling and Simulation Master Plan*, AMSO, 18 May 1995.

Army Modeling and Simulation Office, Standards Category: Object Management, 1997 http://www.amso.army.mil/amso2/sp-div/process/obj-mgt.htm

Army Modeling and Simulation Office *Army Model and Simulation Standards Report FY 97*, December 1996.

Blakely, B., McDonald, K., *Functional Description of the Battle Space White Paper*, United States Army Simulation, Training, and Instrumentation Command, 01 Apr 96.

Hopkins, B., Kastner, K., McCauley, B., *Equipment - Object-oriented Analysis in WARSIM 2000*, 1997 Spring Interoperability Workshop, March, 1997.

JWARS Office, *JWARS\JSIMS Memorandum of Agreement (MOA)*, 30 Nov, 1995, http://www.dtic.mil/jwars/library.html

JWARS Office, *The Joint Warfare System Mission Space Model*, July 10, 1996, http://www.dtic.mil/jwars/library.html

JWARS Office, *The Joint Warfare System Object Model*, September 24, 1996, http://www.dtic.mil/jwars/library.html

Kastner, K., McCauley, B., *Object-oriented Analysis - Case Study*, 1996 Fall Interoperability Workshop, September, 1996.

Larimer. Larry R., *Building an Object Model of A Legacy Simulation*, Naval Postgraduate School, June 1997.

Loral Advanced Distributed Simulation, *ModSAF Kit A (User's Manual)*, DIS Service Center, September, 1995.

Pettitt, Brian L., Rhinesmith, Frank, *Functional Description of the Battle Space*, United States Army Simulation, Training, and Instrumentation Command, http://fdb.orlando.veda.com:443/frames/info.htm

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorenson, W., *Object-oriented Modeling and Design*, Prentice Hall, 1991.

Under Secretary of Defense Memorandum, For: Secretary of the Army, SUBJECT: *DoD High Level Architecture (HLA) for Simulations*, 10 September 1996.

## INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center.....2
   8725 John J. Kingman Road., Ste 0944
   Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library.....2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, California 93943-5101

3. Director Training and Education .....1
   MCCDC, Code C46
   1019 Elliot Rd.
   Quantico, Virginia 22134-5027

4. Director, Marine Corps Research Center .....2
   MCCDC, Code C40RC
   2040 Broadway Street
   Quantico, Virginia 22134-5107

5. Director Studies and Analysis Division .....1
   MCCDC, Code C45
   300 Russel Road
   Quantico, Virginia 22134-5130

6. Marine Corps Representative .....1
   Naval Postgraduate School
   Code 037, Bldg. 234, HA-220
   699 Dyer Road
   Monterey, CA 93940

7. Marine Corps Tactical Systems Support Activity .....1
   Technical Advisory Branch
   Attn: Maj J.C. Cummiskey
   Box 555171
   Camp Pendleton, CA 92055-5080

8. Professor Arnold H. Buss, Code OR/Bu.....2
   Operations Research Dept.
   Naval Postgraduate School
   Monterey, California 93943-5101

9. Director.....1
U. S. Army TRADOC Analysis Center-Monterey
P. O. Box 8692
Monterey, California 93943-0692

10. Major Leroy Jackson.....2
U.S. Army TRADOC Analysis Center-Monterey
P. O. Box 8692
Monterey, California 93943-0692

11. Captain Douglas E. Dudgeon.....2
1817 Leisure World
Mesa, Az 85206

12. Army Modeling and Simulation Office.....4
Attn: Standards and Policy Division, MAJ Johnson
Crystal Gateway 5, Suite 503E
1111 Jefferson Davis Highway
Arlington, Va 22202